NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

VISUALIZING TRANSIENT STRUCTURAL RESPONSE BY EXPANDING SPATIALLY INCOMPLETE TIME HISTORY DATA

by

Scott W. Waltermire

June 1997

Thesis Advisor: Joshua Gordis

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1.AGENCY USE ONLY *(Leave blank)*	2.REPORT DATE	3.REPORT TYPE AND DATES COVERED
	June 1997	Master's Thesis

4.TITLE AND SUBTITLE VISUALIZING TRANSIENT STRUCTURAL RESPONSE BY EXPANDING SPATIALLY INCOMPLETE TIME HISTORY DATA

5.FUNDING NUMBERS

6.AUTHOR(S) Scott Waltermire

7.PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8.PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11.SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a.DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13.ABSTRACT ***(maximum 200 words)***

Due to a limited number of accelerometers available for use, the shock trial for the DDG-51 class destroyer provided a spatially incomplete set of time history data. However, a visualization of the shock response of the entire ship is desired. To this end, finite element model reduction methods are employed to provide a transformation matrix which is used to expand this relatively small collection of data into the same number of degrees of freedom as the finite element model. Using this expanded set of time histories, it is possible to animate the transient response of the structure as a whole.

This approach is investigated using computer-simulated transient response data from a finite element model of a flat plate. The use of static and dynamic reduction methods are explored in the creation of the transformation matrices required for the visualization of the expanded data. The animations are assessed based on a quantitative comparison with the full-order model response.

14. SUBJECT TERMS Finite Element Method

15.NUMBER OF PAGES 80

16.PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19.SECURITY CLASSIFICATION OF ABSTRACT	20.LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

VISUALIZING TRANSIENT STRUCTURAL RESPONSE BY EXPANDING SPATIALLY INCOMPLETE TIME HISTORY DATA

Scott W. Waltermire
Lieutenant, United States Navy
B.S., Unites States Naval Academy, 1991

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1997

ABSTRACT

Due to a limited number of accelerometers available for use, the shock trial for the DDG-51 class destroyer provided a spatially incomplete set of time history data. However, a visualization of the shock response of the entire ship is desired. To this end, finite element model reduction methods are employed to provide a transformation matrix which is used to expand this relatively small collection of data into the same number of degrees of freedom as the finite element model. Using this expanded set of time histories, it is possible to animate the transient response of the structure as a whole.

This approach is investigated using computer-simulated transient response data from a finite element model of a flat plate. The use of static and dynamic reduction methods are explored in the creation of the transformation matrices required for the visualization of the expanded data. The animations are assessed based on a quantitative comparison with the full-order transient model response.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

To gain information concerning the transient response of a complex structure under an arbitrary loading, an analysis of vibration response time history data is required. Unfortunately, a continuous system has an infinite number of degrees of freedom from which data can be extracted. Taking measurements at all of possible locations on the structure is obviously not feasible due to constraints on time, resources, and money. Because of these constraints, accelerometers are placed at a few specific locations on a structure and the data is then analyzed. While this information is quite useful, the collection of data still does not provide a clear picture of the response at the many locations where data was not taken. More importantly, since each piece of data is for a specific location, it takes quite a bit of imagination to get an understanding of how the structure as a whole is responding to the loading.

Due to the limits on the number of accelerometers available, the shock trial conducted on the DDG-51 class destroyer provided a spatially incomplete set of test data. In addition to the information collected during the test, knowledge of the transient response at locations on the ship where data was not taken was desired. By expanding this collection of test data into a spatially complete set, the transient response could be visualized in a computer simulation.

This thesis investigates the strengths and weaknesses of expanding spatially incomplete test data using finite element model reduction methods. These reduction methods are used to create a transformation matrix which makes it possible to expand a relatively small collection of simulated test data into a set that corresponds to a much larger number of degrees of freedom.

The transformation matrix is created from a finite element model of the structure whose system mass and stiffness matrices have been partitioned in accordance with the locations of the accelerometers on the actual structure. The final collection of data becomes the time histories of all the coordinates in the full-order finite element model and has the same number of degrees of freedom.

The full-order set of data is then animated to create a real-time visualization of the dynamic response of the structure as a continuous system. Every column in the expanded matrix is a representation of the dynamics of each node in the finite element model at a specific time interval. With this information, a very clear understanding of the vibration response of a complex structure is possible.

There are two different reduction methods used in this thesis to create the transformation matrices. The first is the static, or Guyan, reduction and the second is the Improved Reduced System (IRS) reduction. The IRS method, while initially more computationally expensive, provides an improvement because it approximates the inertia forces that are present in a dynamic problem. A sensitivity analysis for each simulation is provided to highlight the accuracy of both the static and the dynamic reduction methods.

II. THEORY

A. MATRIX PARTITIONING

In the physical coordinate system, the expansion of the collected time history data can be expressed in the following way:

$$\begin{Bmatrix} \mathbf{x}_a \\ \mathbf{x}_o \end{Bmatrix} = \begin{bmatrix} \mathbf{I} \\ \mathbf{T} \end{bmatrix} \{\mathbf{x}_a\} \tag{1}$$

where $\{\mathbf{x}_a\}$ is a vector containing the dynamic response data collected from the structure, $\{\mathbf{x}_o\}$ is the response at coordinates other than the data collection points, and [T] is a non-square transformation matrix used in the expansion of $\{\mathbf{x}_a\}$ to the full-order response set, and [I] is the identity matrix.

The subscript 'a' refers to the *analysis* set of model coordinates, which from here on will be called simply the 'a-set'. This is the set of coordinates in the model that corresponds to the locations on the structure where data was taken. Conversely, the subscript 'o' refers to the *omitted* set. This 'o-set' refers to all subsequent model coordinates where data is *not* taken. By operating on the 'a-set' time history data set with the proper transformation matrix, the 'o-set' dynamic response is calculated.

Generally, due to the size of complex models, the a-set system will be much smaller than the o-set system. The model of a complex structure will normally contain a huge number of degrees of freedom while the number of accelerometers is relatively small in comparison.

B. STATIC TRANSFORMATION MATRIX

By partitioning the stiffness relation, $\mathbf{f} = \mathbf{kx}$, the static transformation matrix can easily be found. The equation becomes:

$$\begin{bmatrix} \mathbf{K}_{aa} & \mathbf{K}_{ao} \\ \mathbf{K}_{oa} & \mathbf{K}_{oo} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_a \\ \mathbf{x}_o \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_a \\ \mathbf{f}_o \end{Bmatrix} \tag{2}$$

By taking care to partition the equation such that there are no excitations applied at the o-set coordinates, then $\{\mathbf{f}_o\}$ can be set equal to {0}. Now the relation between the o-set and a-set response can be expressed as:

$$\{\mathbf{x}_o\} = \left[-\mathbf{K}_{oo}^{-1}\mathbf{K}_{oa}\right]\{\mathbf{x}_a\} \tag{3}$$

The static transformation matrix is then [Ref. 1],

$$\mathbf{T}_{stat} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_{oo}^{-1}\mathbf{K}_{oa} \end{bmatrix} \tag{4}$$

C. IRS TRANSFORMATION MATRIX

The derivation of the IRS reduction transformation matrix begins with the partitioned equation of motion for a vibrating system [Ref. 2]:

$$\begin{bmatrix} \mathbf{M}_{aa} & \mathbf{M}_{ao} \\ \mathbf{M}_{oa} & \mathbf{M}_{oo} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{x}}_a \\ \ddot{\mathbf{x}}_o \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_{aa} & \mathbf{K}_{ao} \\ \mathbf{K}_{oa} & \mathbf{K}_{oo} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_a \\ \mathbf{x}_o \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_a \\ \mathbf{f}_o \end{Bmatrix} \tag{5}$$

Assuming a harmonic motion of frequency, Ω, the acceleration term in equation (5) can be replaced with $\ddot{\mathbf{x}} = -\Omega^2\mathbf{x}$. Again partitioning such that there are no excitations at the o-set coordinates, we arrive at the exact structural dynamics reduction relation [Ref. 3]:

$$\{\mathbf{x}_o\} = \left[\mathbf{I} - \Omega^2 \mathbf{K}_{oo}^{-1}\mathbf{M}_{oo}\right]^{-1}\left[-\mathbf{K}_{oo}^{-1}\mathbf{K}_{oa} + \Omega^2 \mathbf{K}_{oo}^{-1}\mathbf{M}_{oa}\right]\{\mathbf{x}_a\} \tag{6}$$

(a) (b)

Ideally, the desired transformation matrix will be frequency independent and Eq. (6) is dependent on the driving frequency by the Ω^2 term. Truncating the binomial expansion of Eq. (6a) after the terms that include Ω^2 yields:

$$\{\mathbf{x}_o\}\left[-\mathbf{K}_{oo}^{-1}\mathbf{K}_{oa} + \Omega^2(\mathbf{K}_{oo}^{-1}\mathbf{M}_{oa} - \mathbf{K}_{oo}^{-1}\mathbf{M}_{oo}\mathbf{K}_{oo}^{-1}\mathbf{K}_{oa})\right]\{\mathbf{x}_a\} \tag{7}$$

The frequency dependency is then eliminated by the following approximation of the acceleration:

$$\Omega^2\{\mathbf{x}_a\} = \left[\mathbf{M}_{stat}\right]^{-1}\left[\mathbf{K}_{stat}\right] \tag{8}$$

where the statically reduced mass and stiffness matrices are given by the following [Ref. 2,3]:

$$\mathbf{M}_{stat} = \mathbf{T}_{stat}^T \mathbf{M}\mathbf{T}_{stat} \tag{9a}$$

$$\mathbf{K}_{stat} = \mathbf{T}_{stat}^T \mathbf{K}\mathbf{T}_{stat} \tag{9b}$$

Substitution of Eq. (8) into Eq. (7) provides:

$$\{\mathbf{x}_o\}\left[-\mathbf{K}_{oo}^{-1}\mathbf{K}_{oa} + \left[\mathbf{K}_{oo}^{-1}\mathbf{M}_{oa} - \mathbf{K}_{oo}^{-1}\mathbf{M}_{oo}\mathbf{K}_{oo}^{-1}\mathbf{K}_{oa}\right]\left[\mathbf{M}_{stat}\right]^{-1}\left[\mathbf{K}_{stat}\right]\right]\{\mathbf{x}_a\} \tag{10}$$

which is the IRS reduction relationship between the a-set and o-set

coordinates. The IRS reduction transformation matrix can then be written as [Ref. 2,3]:

$$\mathbf{T}_{IRS} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_{oo}^{-1}\mathbf{K}_{oa} + \left[\mathbf{K}_{oo}^{-1}\mathbf{M}_{oa} - \mathbf{K}_{oo}^{-1}\mathbf{M}_{oo}\mathbf{K}_{oo}^{-1}\mathbf{K}_{oa}\right]\mathbf{M}_{stat}^{-1}\mathbf{K}_{stat} \end{bmatrix} \tag{11}$$

D. VISUALIZATION MATRIX

After operating on $\{\mathbf{x}_a\}$ with the transformation matrix, the response for the entire structure is partitioned in the form:

$$\{\mathbf{x}(t)\} = \begin{Bmatrix} \mathbf{x}_a(t) \\ \mathbf{x}_o(t) \end{Bmatrix} = \begin{bmatrix} \{\mathbf{x}_a(t_1)\} & . & . & \{\mathbf{x}_a(t_n)\} \\ \{\mathbf{x}_o(t_1)\} & . & . & \{\mathbf{x}_o(t_n)\} \end{bmatrix} \tag{12}$$

where the columns in the matrix are a representation of the dynamic response at each node in the model at a specific instant in time. However, this partitioning does not correspond to the nodes in the model, therefore Eq. (12) must again be partitioned such that it is in line with the nodes in the model. To get a real-time visualization of the structure, simply march through the matrix taking snapshots of the model at each time interval.

III. FINITE ELEMENT MODEL FORMULATION

The finite element method used in the creation of the plate model for this thesis is based on the shear deformable displacement formulation. The element chosen is the four-noded quadrilateral plate element. Each node in the model has three degrees of freedom which are the transverse displacement, w, and the two rotations about the midplane, θ_x and θ_y. Using the following bilinear isoparametric shape functions [Ref. 4],

$$H_1(\xi,\eta) = \frac{1}{4}(1-\xi)(1-\eta) \tag{13a}$$

$$H_2(\xi,\eta) = \frac{1}{4}(1+\xi)(1-\eta) \tag{13b}$$

$$H_3(\xi,\eta) = \frac{1}{4}(1+\xi)(1+\eta) \tag{13c}$$

$$H_4(\xi,\eta) = \frac{1}{4}(1-\xi)(1+\eta) \tag{13d}$$

the transverse displacement and slopes are interpolated as:

$$w = \sum_{i=1}^{n} H_i(\xi,\eta) w_i \tag{14a}$$

$$\theta_x = \sum_{i=1}^{n} H_i(\xi,\eta)(\theta_x)_i \tag{14b}$$

$$\theta_y = \sum_{i=1}^{n} H_i(\xi,\eta)(\theta_y)_i \tag{14c}$$

These isoparametric shape functions are defined in terms of a normalized domain $-1 \le \xi \le 1$ and $-1 \le \eta \le 1$.

A. INTEGRATION TECHNIQUE

As previously stated, the shape functions used in this model are for the

bilinear isoparametric element. This type of element was chosen in order to make the finite element model general in the sense that it can be applied to any plate geometry. The elements in the physical coordinate system are mapped into the isoparametric coordinate system by the following relations:

$$x = \sum_{i=1}^{4} H_i(\xi,\eta)x_i \tag{15a}$$

$$y = \sum_{i=1}^{4} H_i(\xi,\eta)y_i \tag{15b}$$

where x and y are physical coordinates corresponding to the nodes in the element. Gauss-Legendre quadrature is used to perform all integrations.

B. ELEMENT MATRIX

1. Elemental Stiffness Matrix

Without providing the derivation, the elemental stiffness matrix, $[\mathbf{K}^e]$, for plate bending is [Ref. 4]:

$$[\mathbf{K}^e] = \frac{h^3}{3}\int_{\Omega^e} \mathbf{B}_b^T \mathbf{D}_b \mathbf{B}_b d\Omega + kh\int_{\Omega^e} \mathbf{B}_s^T \mathbf{D}_s \mathbf{B}_s d\Omega \tag{16}$$

where h is the thickness of the plate, k equals $\frac{5}{6}$ and is the shear energy correction factor, Ω^e is the two dimensional element domain, and

$$[\mathbf{B}_b] = \begin{bmatrix} \frac{\partial H_1}{\partial x} & 0 & 0 & \frac{\partial H_2}{\partial x} & 0 & 0 & \frac{\partial H_3}{\partial x} & 0 & 0 & \frac{\partial H_4}{\partial x} & 0 & 0 \\ 0 & \frac{\partial H_1}{\partial y} & 0 & 0 & \frac{\partial H_2}{\partial y} & 0 & 0 & \frac{\partial H_3}{\partial y} & 0 & 0 & \frac{\partial H_4}{\partial y} & 0 \\ \frac{\partial H_1}{\partial y} & \frac{\partial H_1}{\partial x} & 0 & \frac{\partial H_2}{\partial y} & \frac{\partial H_2}{\partial x} & 0 & \frac{\partial H_3}{\partial y} & \frac{\partial H_3}{\partial x} & 0 & \frac{\partial H_4}{\partial y} & \frac{\partial H_4}{\partial x} & 0 \end{bmatrix} \tag{17}$$

$$[\mathbf{B}_s]=\begin{bmatrix} -H_1 & 0 & \frac{\partial H_1}{\partial x} & -H_2 & 0 & \frac{\partial H_2}{\partial x} & -H_3 & 0 & \frac{\partial H_3}{\partial x} & -H_4 & 0 & \frac{\partial H_4}{\partial x} \\ 0 & -H_1 & \frac{\partial H_1}{\partial y} & 0 & -H_2 & \frac{\partial H_2}{\partial y} & 0 & -H_3 & \frac{\partial H_3}{\partial y} & 0 & -H_4 & \frac{\partial H_4}{\partial y} \end{bmatrix} \tag{18}$$

The constitutive equation for the plane stress condition is:

$$[\mathbf{D}_b]=\frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{19}$$

while the constitutive equation for shear is,

$$[\mathbf{D}_s]=\begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix} \tag{20}$$

G is the shear modulus, E is the modulus of elasticity, and ν is Poisson's ratio. One thing of importance to be noted about Eq. 15 is that the shear energy becomes dominant over the bending energy as the thickness of the plate becomes small relative to the side length. This is called *shear-locking*. To account for this problem, a reduced integration technique is used. The bending term is integrated exactly using 2x2 Gauss-Legendre quadrature while the shear term is under-integrated using 1x1 Gauss-Legendre quadrature.

2. Elemental Mass Matrix

The elemental mass matrix is found by simply integrating the properly weighted shape functions. The general equation for determining this matrix is:

$$\left[\mathbf{M}^{e}\right]=\int_{\Omega^{e}}\rho A\begin{Bmatrix}H_1\\H_2\\H_3\\H_4\end{Bmatrix}\begin{bmatrix}H_1 & H_2 & H_3 & H_4\end{bmatrix}d\Omega \tag{21}$$

In Eq. (19), ρ is the density of the plate element, A is the elemental area, and $[H]$ is the same shape functions as in Eq. (12). 2x2 Gauss-Legendre quadrature is used to perform all integrations for the element mass matrix.

IV. COMPUTER SIMULATIONS

The investigation of expanding time history data was conducted with the aid of computer simulations. The dynamic response at each node of a plate finite element model was used to simulate the 'actual' solution. From this baseline model, an a-set system of time histories was chosen which is assumed to be the data taken from an actual structure. The system matrices in the finite element model were then partitioned in order to create the transformation matrix required to expand the a-set time histories to full-order. By comparing the dynamic response of the finite element model to the response found by expanding the a-set system, the validity of the process could be studied and verified.

A. EXAMPLE (1): TWO DOF IN THE A-SET

The initial simulations are conducted on a square, flat plate (see Figure 1) comprised of twenty-five elements and thirty-six nodes. Degrees of freedom 42 and 87 are assumed to be the 'actual' response and therefore make up the a-set system. Conversely, the o-set system is comprised of all other DOF. The two DOF in the a-set system correspond to the transverse motion at nodes 14 and 29. The external force is a blast function (see Figure 2) applied at node twenty-one. Each external spring has a stiffness equal to 1000 lb-in and is attached to the ground.

Two separate simulations are performed using this plate and a-set system. The first simulation uses the static transformation matrix to expand

the a-set and the second uses the IRS. Both are then compared to the full-order plate response. Unfortunately, it is impossible to show the visualization on paper so the comparison will be on a per DOF basis.

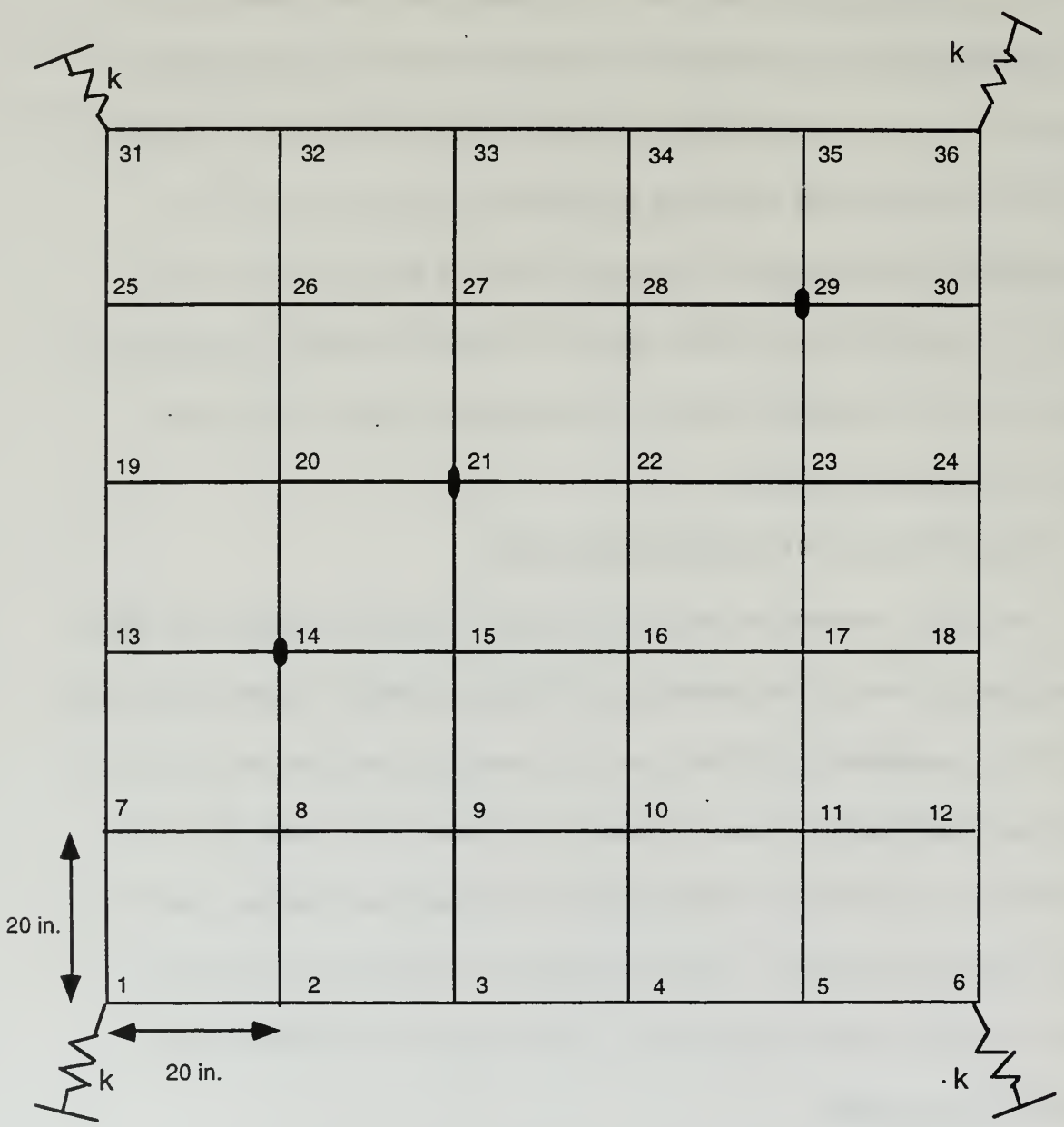

Figure 1 Plate Model

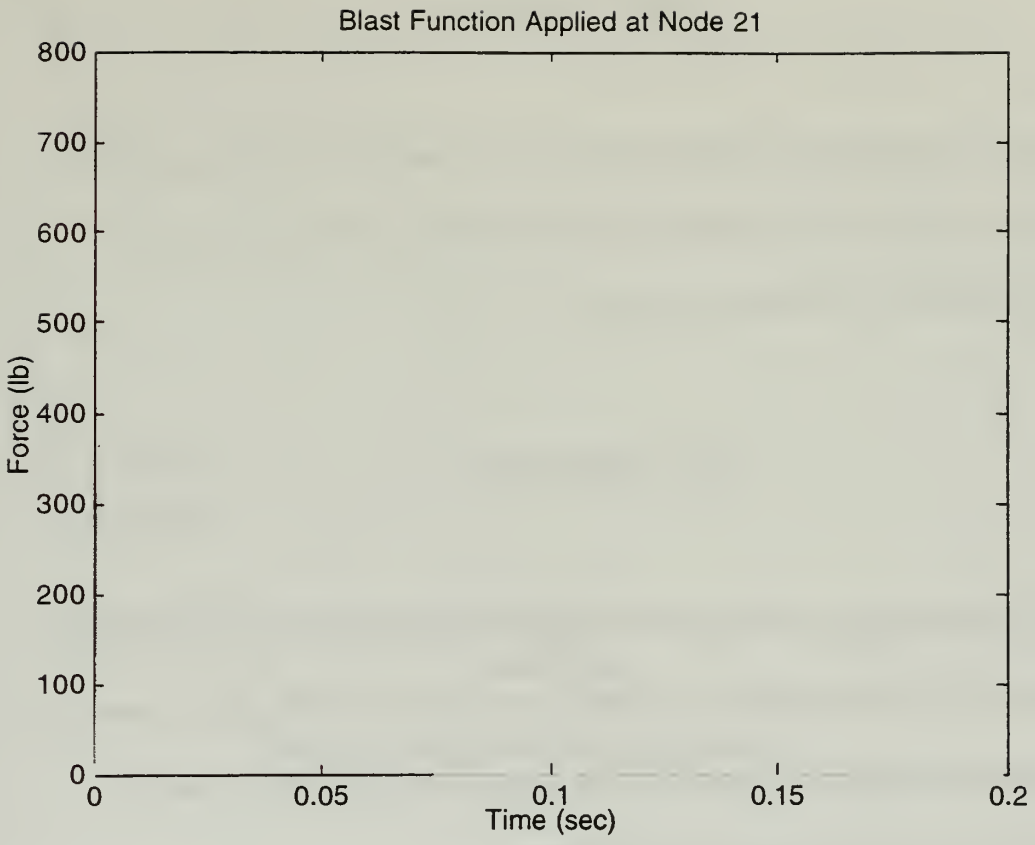

Figure 2 Forcing Function

1. Full-Order Plate Results

The dynamic response of the full-order plate model was solved by transforming the general equation of motion,

$$[M]\{\ddot{x}\}+[K]\{x\}=\{f(t)\} \tag{22}$$

into modal coordinates by the following relation:

$$\{x(t)\} = [\phi]\{q(t)\} \tag{23}$$

in which $\{q(t)\}$ is the modal displacements (as a function of time) and $[\phi]$ is a matrix of mass normalized modeshapes found from solving the typical structural dynamics eigenvalue problem:

$$\left[\mathbf{K} - \omega_n^2 \mathbf{M}\right]\{\phi\} = \{0\} \tag{24}$$

The modal displacements are found by numerically solving a convolution integral [Ref. 5]. The damping in the plate is assumed to be 2% and is inserted into the integral calculation. Once $\{q(t)\}$ is found, Eq. (21) is then used to convert back to the physical coordinate system.

Now, the time history at every node in the plate is known. These results will be used to compare the accuracy of the static and dynamic expansions of the a-set.

2. Static Transformation Results

Figures III, IV, and V are per-node samples of the full-order plate model response in comparison to the response found by the static expansion of the 2 DOF a-set. It is clear that the static transformation will provide an adequate approximation of the 'true' response at certain nodes.

Unfortunately, Figure 3 proves that the difference between the 'true' solution and the expanded set can be greater than 50%.

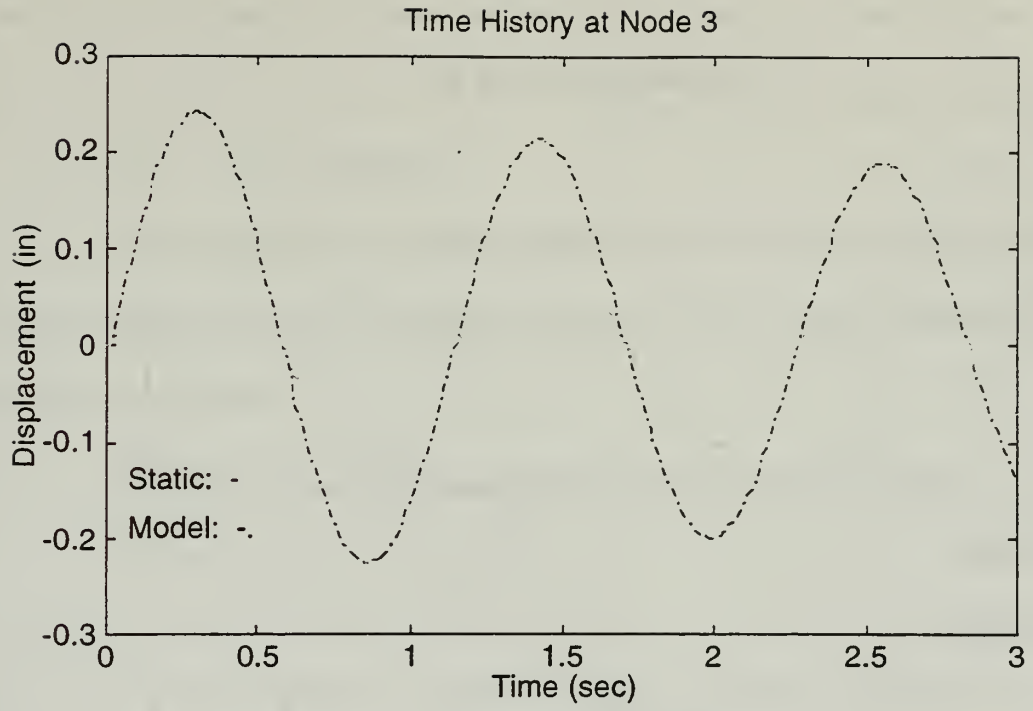

Figure 3 Static Response at Node 3

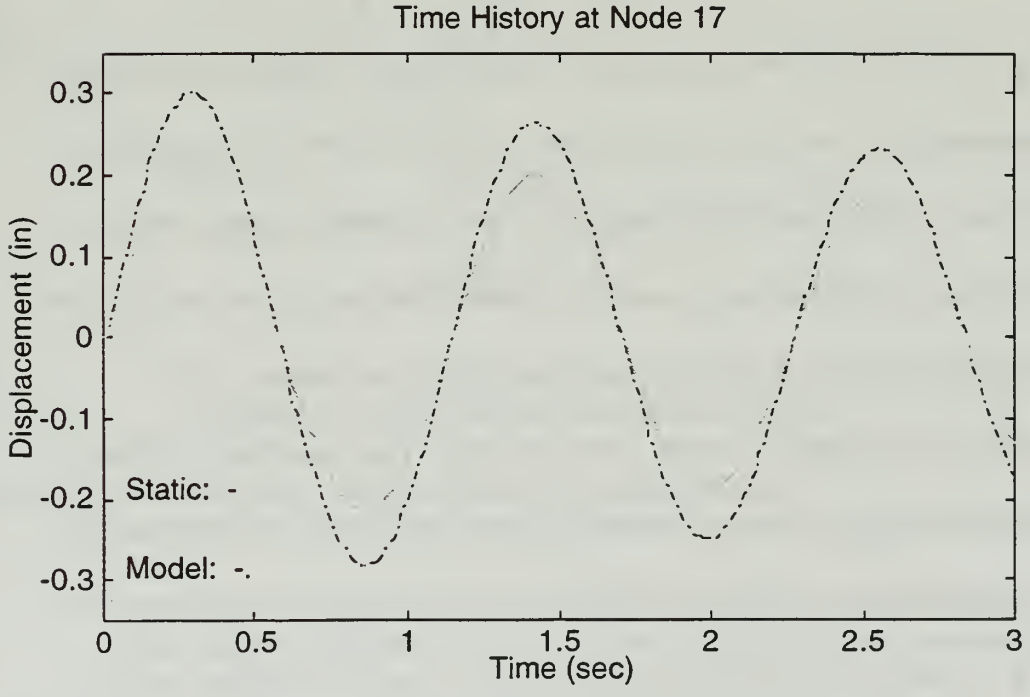

Figure 4 Static Response at Node 17

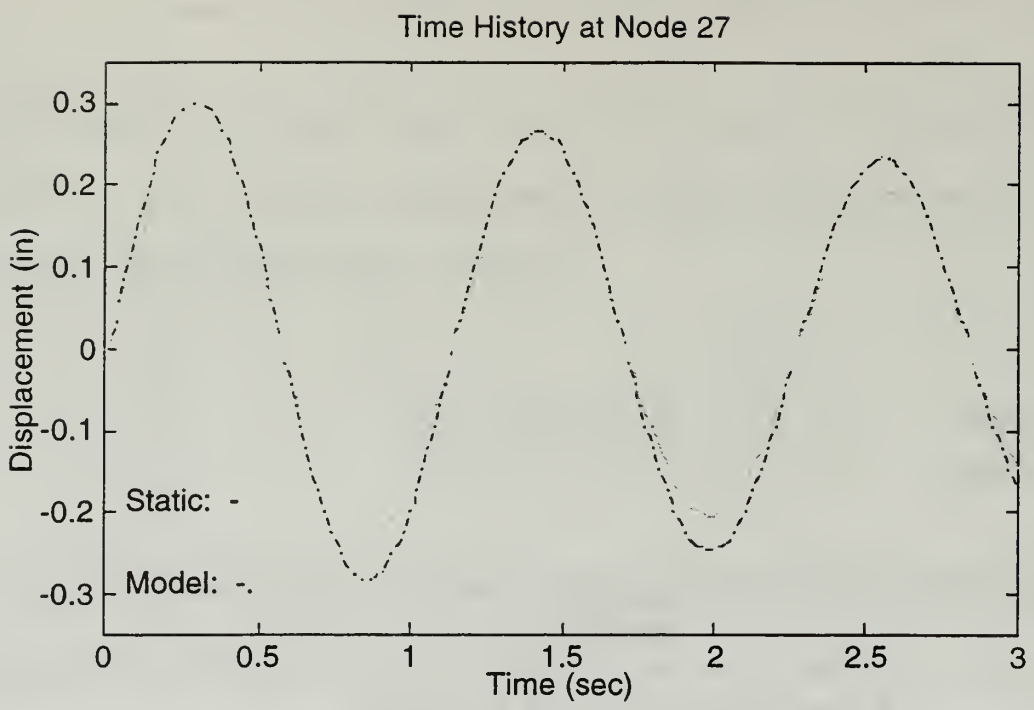

Figure 5 Static Response at Node 27

Having errors of this magnitude can lead to very misleading results in the final visualization. It will provide an idea of how the structure vibrates but the animation is distorted and skewed. If a more accurate representation is the requirement, then other steps must be taken. Obviously, one way to get a more accurate solution is to increase the size of the a-set system. An example solution of the larger a-set is provided later. The drawback to this is more data must be taken which is oftentimes impossible due to limits on the number of accelerometers available. Another way to change the accuracy of the solution is to alter the locations where the data is actually taken on the structure. An example is also provided later in this chapter of greatly

decreasing the accuracy of the static solution by choosing an a-set system badly.

a. Error Analysis

To understand the differences in the full-order and static plate solutions without providing the time histories at every node, a sensitivity analysis was conducted.

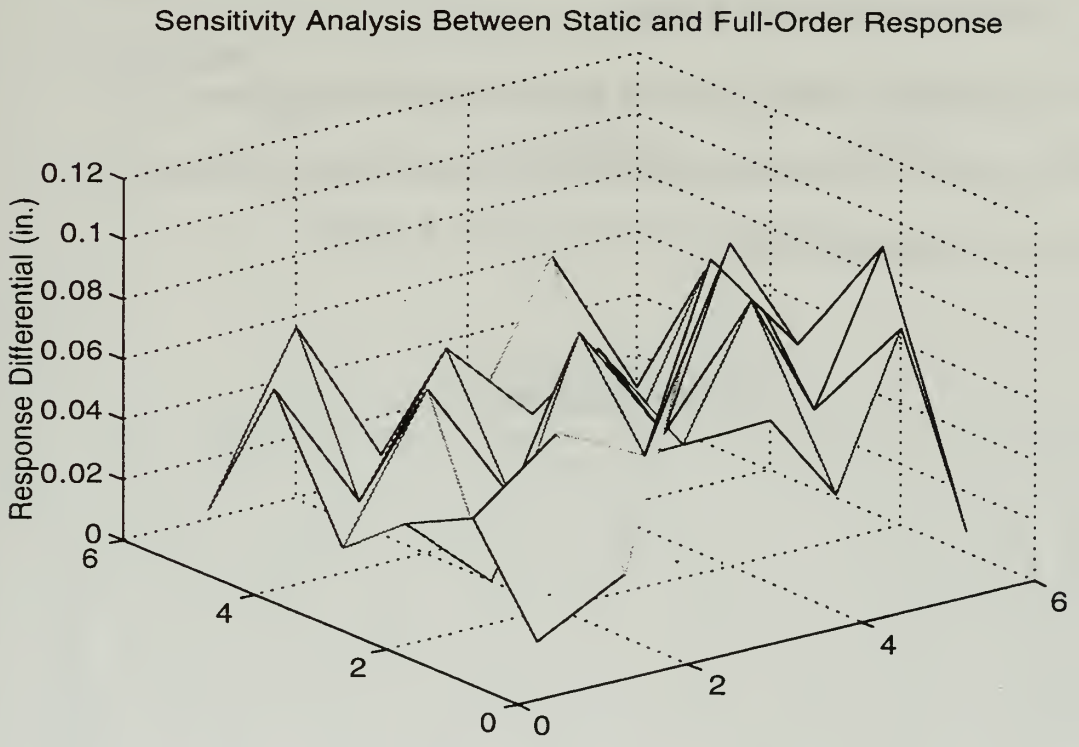

Figure 6 Error of Static Reduction Method

note: Figure is orientated such that the 0/0 point is node 1

The largest difference between the full-order plate response and the expanded time history set was found and plotted. This analysis is essentially a 'worst-case' scenario because the code searched through time to

find the absolute maximum error at each node and these errors did not always occur at the same instant in time.

The maximum differential appears to be approximately 0.1 inches, but the movement of the plate is only about 0.3 inches. Under these circumstances, an error of this magnitude is quite significant, and therefore the static reduction probably should not be used.

3. IRS Transformation Results

Using the existing plate model and forcing function (see Figures 1 and 2), the solution for the IRS expanded data is provided as before. The same nodes are given as samples here:

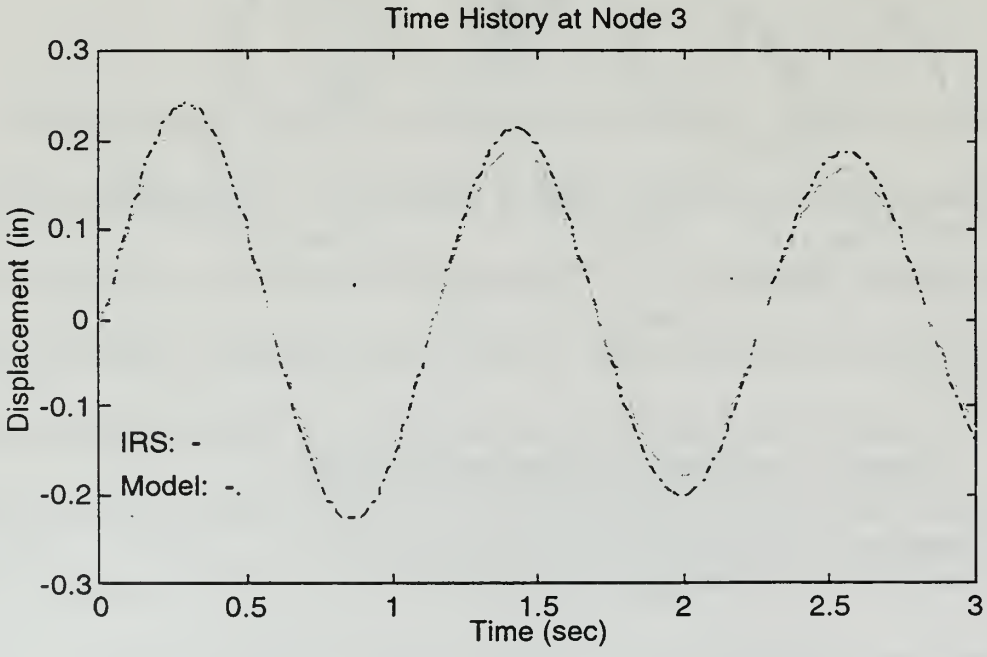

Figure 7 IRS Response at Node 3

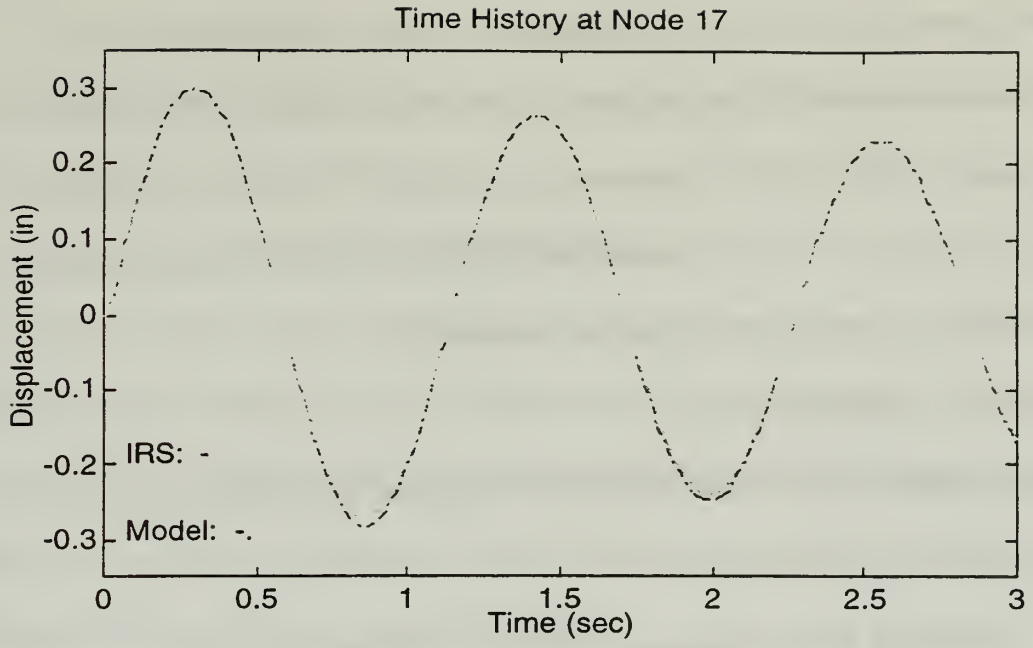

Figure 8 IRS Response at Node 17

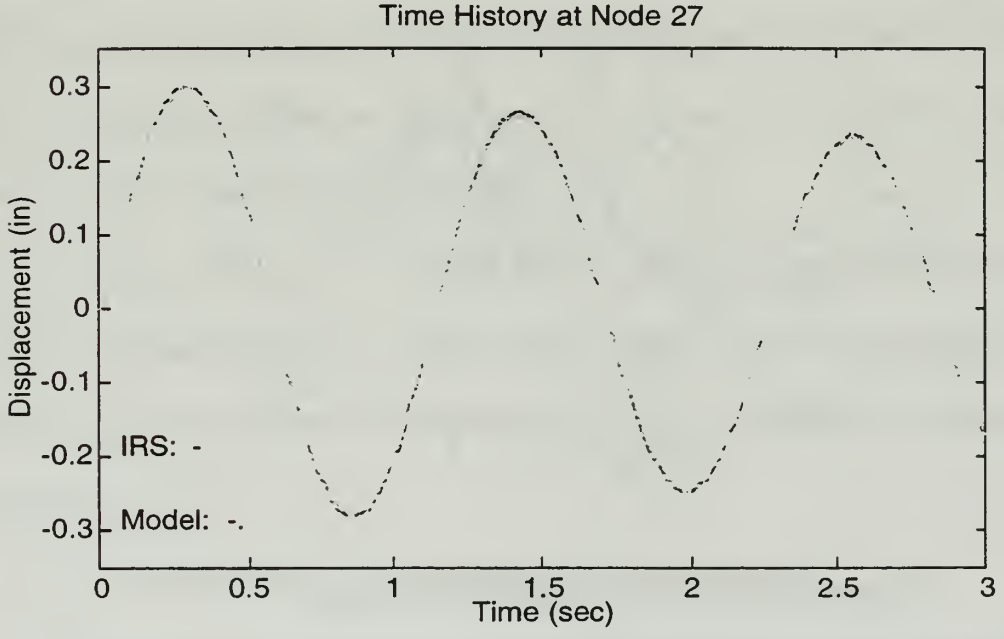

Figure 9 IRS Response at Node 27

These figures show that the IRS expanded data is much more accurate in comparison to the static transformation. Even with an a-set of only 2 DOF out of the possible 108, the IRS transformation provides a very good solution. Clearly, the correction in the IRS transformation matrix due to inertial forces will improve the accuracy of the solution immensely.

a. Error analysis

A sensitivity analysis for the IRS expanded data is also provided:

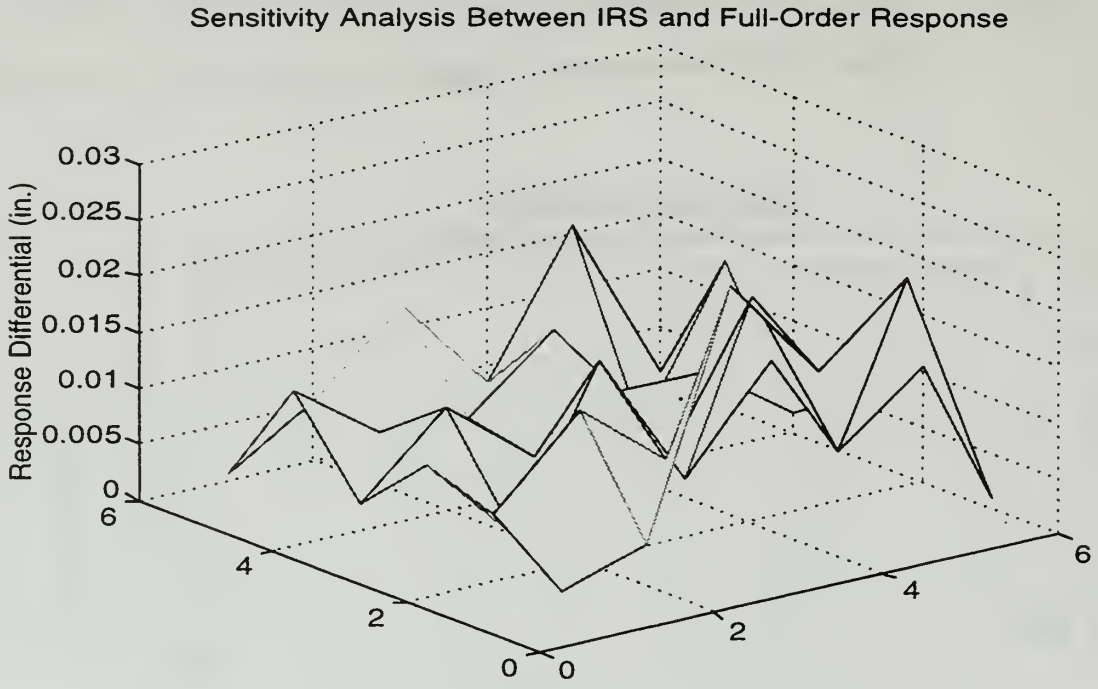

Figure 10 Error of IRS Reduction Method

note: Figure is orientated such that the 0/0 point is node 1

The maximum difference in Figure 10 is less than 0.02 inches. This accuracy is an order of magnitude better than the static reduction. The

approximation of the inertial forces inherent in the IRS transformation matrix appears to work quite well.

B. EXAMPLE (2): FOUR DOF IN THE A-SET

This example shows that the accuracy in both solutions can be greatly improved by increasing the size of the a-set. By merely increasing the size of the a-set by two, it will be seen that the improvement in accuracy is at least an order of magnitude. As mentioned earlier, although desirable, increasing the number of data points may not be feasible.

For this simulation, the forcing function used is the same as in Figure 2. The plate model, shown in Figure 1, is changed such that the data is instead extracted at nodes 8, 11, 26, and 29. In the transverse direction, these nodes correspond to degree of freedom 24, 33, 78, and 87. All other variables in the simulation are unchanged.

On a per node basis, the results are very hard to comprehend because the graphs are almost exactly on top of each other. For the IRS expanded data, no difference can be detected. Because of this, only the sensitivity analysis is provided.

1. IRS and Static Transformation Error Analysis

By viewing these graphs, it becomes clear how the accuracy can by greatly increased by adjusting the number of DOF in the a-set system. The greatest difference in inches of the response for the static solution is 0.01.

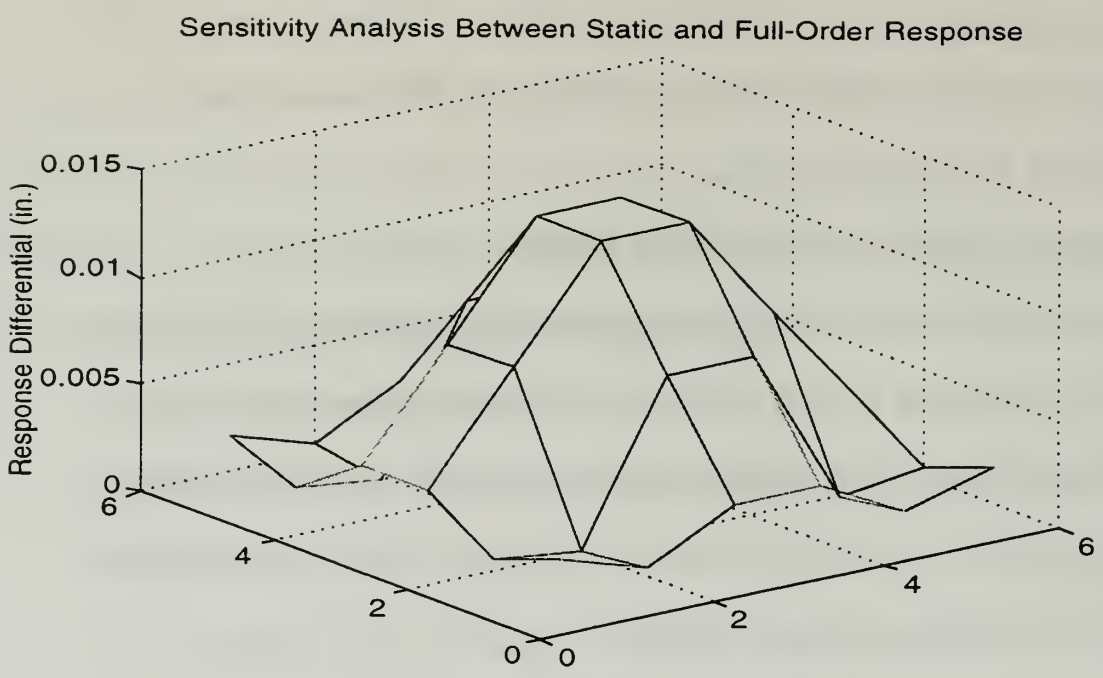

Figure 11 Error of Static Reduction Method

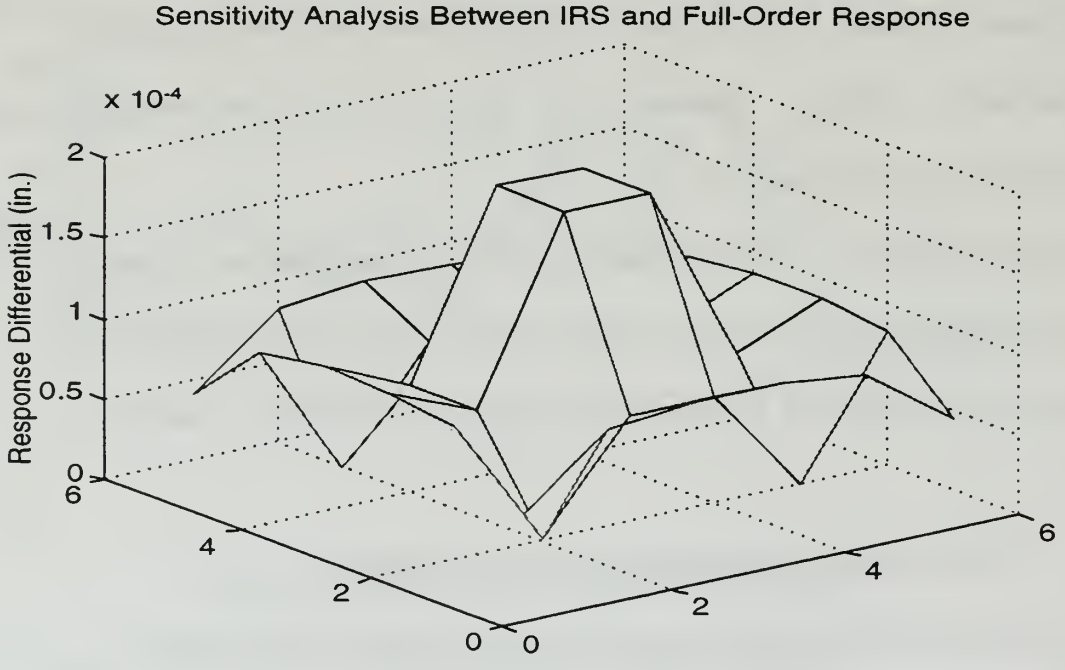

Figure 12 Error of IRS Reduction Method

note: Figures are orientated such that the 0/0 point is node 1

The difference for the IRS solution is much smaller at 0.00015 in. As expected, the IRS reduction method remains the more accurate of the two, but both methods provide very good solutions. If the visualization could be shown here, one would not be able to discern a difference between the two animations and the 'true' solution.

C. EXAMPLE (3): A-SET INCLUDES CONSTRAINED NODES

For this simulation, the accelerometers are assumed to be located at nodes which are constrained to ground through external springs. For the plate, these nodes are numbered 1, 6, 31, and 36. The corresponding DOF's in the a-set system are 3, 18, 93, and 108, respectively.

1. Static Transformation Results

Figures 13 and 14 are a sampling of the results from this particular simulation and they show a very interesting situation to consider. These time history samples clearly indicate that the static transformation matrix will not provide an accurate solution for this particular a-set. On closer inspection of the two previous graphs, it is seen that the static transformation provided the exact same solution at nodes 10 and 29. In fact, although not shown, this is the exact response calculated at every node in the plate, and, when placed in animation, the plate exhibits rigid body motion in the transverse direction.

The reason for this rigid body motion can be attributed to the location of the a-set system. By taking time history data at only those nodes constrained to ground through springs, the transformation matrix imparts

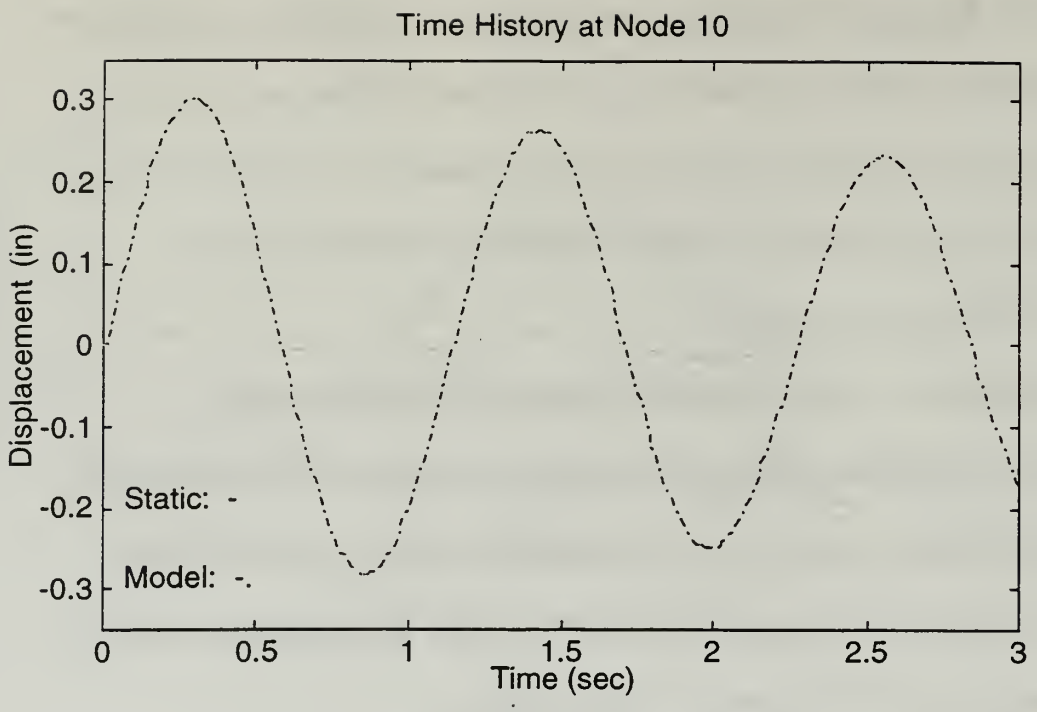

Figure 13 Static Response at Node 10

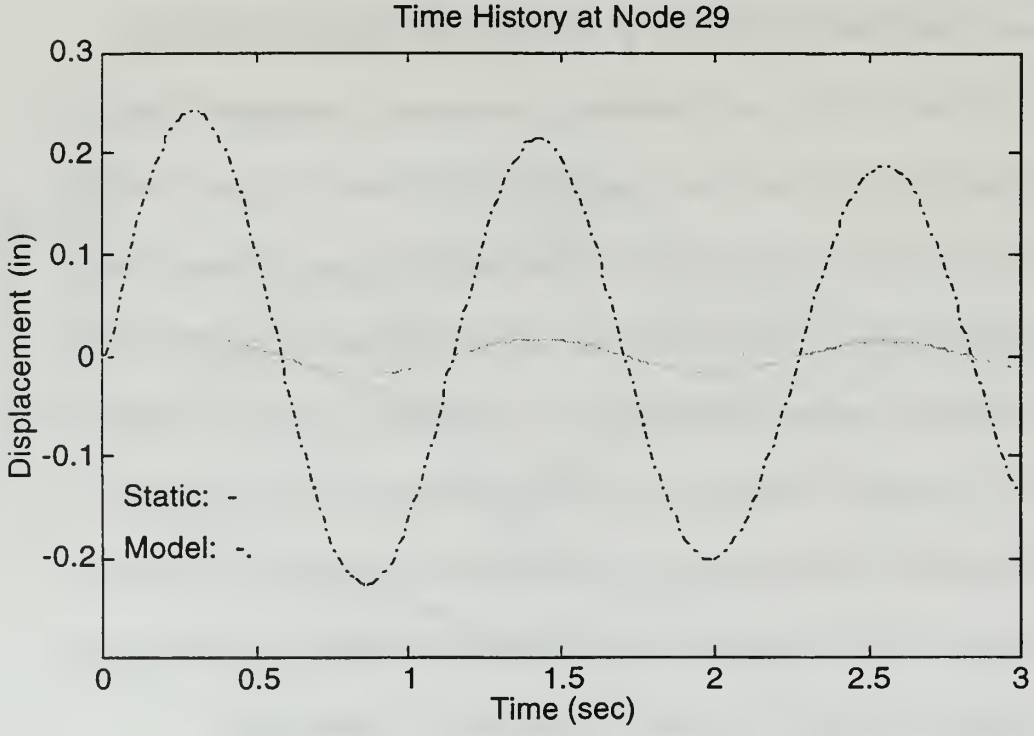

Figure 14 Static Response at Node 29

the strain energy of the springs to the system and ignores any of the plate's internal strain energy. In other words, the response calculated in the o-set is effectively 'zeroed out' by the transformation matrix. By discarding the motion of the individual o-set nodes, the plate moves with the springs as a rigid body. A detailed explanation of this behavior is provided in the next chapter.

a. Error analysis

The inaccuracy of the solution becomes clear with the sensitivity analysis.

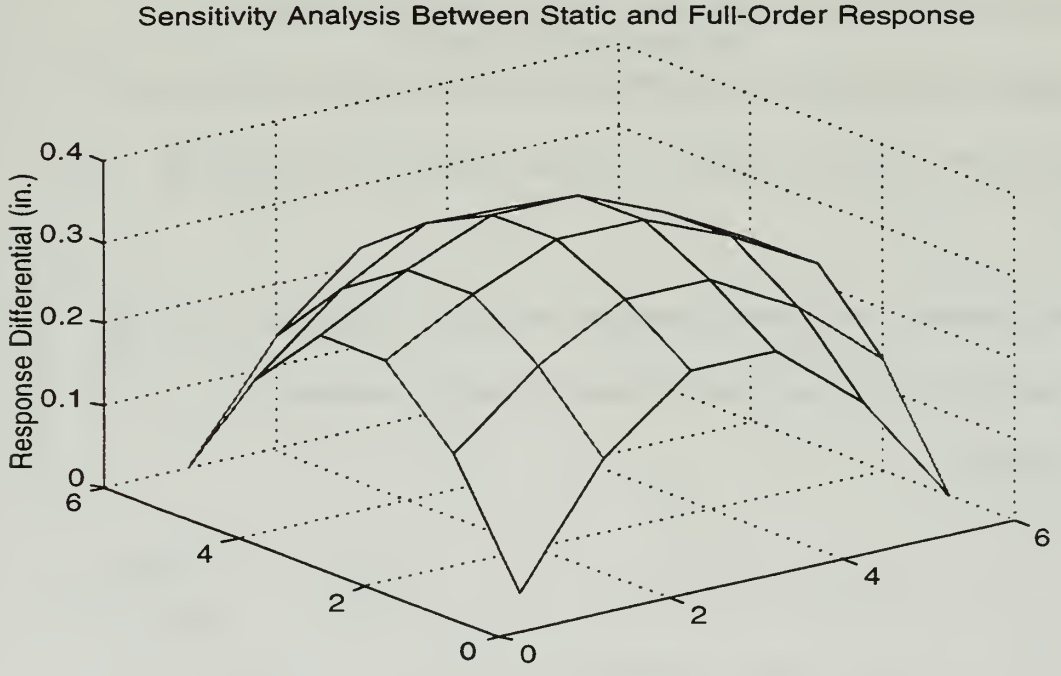

Figure 15 Error of Static Reduction Method

note: Figure is orientated such that the 0/0 point is node 1

Obviously, rigid body motion is an inaccurate solution to this situation. As expected, the nodes at the four corners exhibit no error while the center of the plate has the greatest. From Figure 13, the maximum amplitude of the plate motion is about 0.3 in. which is the same as the maximum error shown here. Example (2) proved that a very accurate solution can be found using the static reduction method when the a-set contains four DOF. This example shows that care must be taken when using the static reduction method that the accelerometers are not placed at constrained nodes.

2. IRS Transformation Results

The problems using the static reduction method are not exhibited by the IRS method. Due to the correction in the derivation of the transformation matrix due to the inertial forces, the motion of the plate is taken into account. For this reason, the actual response of the plate is calculated.

The following time history samples prove the usefulness of the IRS reduction method when data must be taken at nodes constrained to ground through a spring:

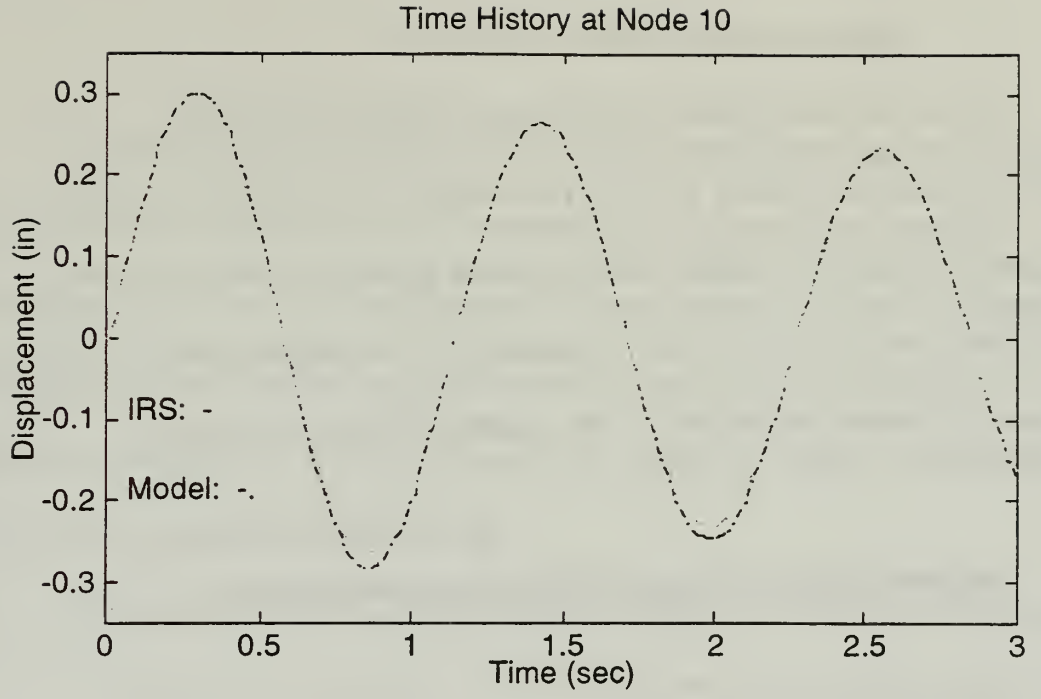

Figure 16 IRS Response at Node 10

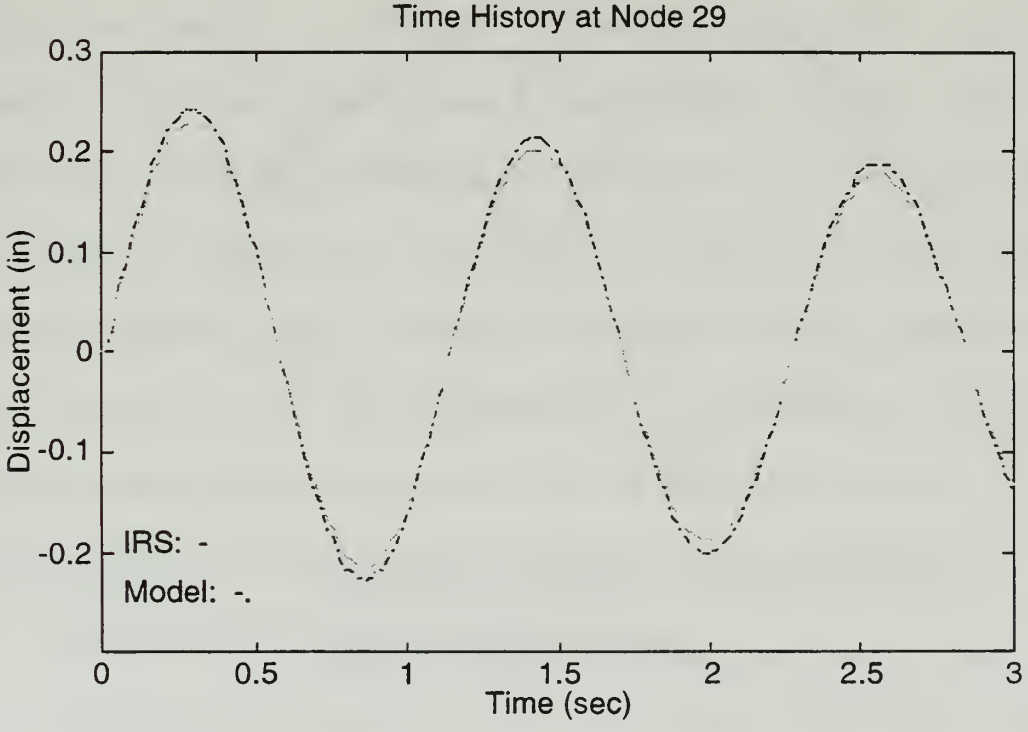

Figure 17 IRS Response at Node 29

a. **Error analysis**

Figure 18 has the same basic shape as Figure 15, but the maximum error this time is only 0.02 in. The inertial force correction to the IRS transformation matrix enables a very accurate solution to be found even though the strain energy of the plate is ignored. For this reason, using the IRS method allows for fewer limitations in the location of the a-set system.

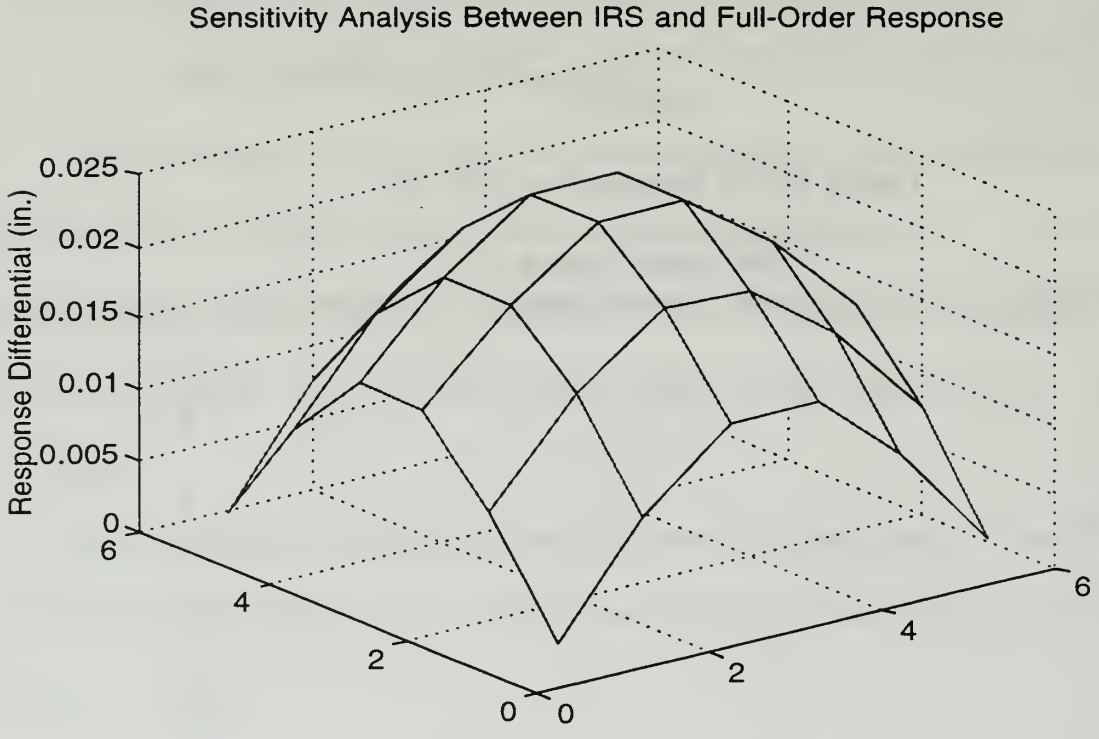

Figure 18 Error of IRS Reduction Method

note: Figures are orientated such that the 0/0 point is node 1

V. LESSONS LEARNED

The three examples provided in the previous chapter are only a few of the simulations conducted in the investigation of expanding spatially incomplete time histories to full-order. Several other situations requiring investigation were checked and will be addressed here. This chapter highlights the lesson's learned from all the computer simulations without providing the detail seen previously.

A. ACCURACY DEPENDS UPON THE NUMBER OF DOF IN THE A-SET

As expected, the number of DOF in the a-set system greatly affects the accuracy of the solution. The increase in accuracy seen in changing the a-set system from Example (1) to Example (2) is excellent proof of this. Actually, a dramatic improvement could be seen by increasing the number of DOF in the a-set to three. Interestingly, independent of the size, shape, or number of DOF in the model, a system of only one DOF in the a-set did not provide a viable solution for either method. Therefore the minimum number of DOF in the a-set is restricted to two. Simulations were not run for situations where the number is greater than four because, as seen in Example (2), an a-set containing four DOF expands into a solution that is essentially exact.

B. LOCATION OF THE A-SET IS CRITICAL

In addition to the number of DOF, the accuracy of the solution can be greatly affected by the location of the a-set. Two general cases have been found that should be taken into account. The first is to pick an a-set location that is

certain to impart strain energy into the transformation matrix in order to avoid the rigid-body modes. This restriction is of vital importance if the static reduction is chosen. The second is to ensure that the time history data is taken at points 'distant' from one another.

Example (3) revealed a problem with the static reduction method when the nodes chosen for the a-set are constrained. As mentioned earlier, the reason is that the only strain energy imparted into the system is the stiffness of the springs while the internal strain energy of the plate is 'zeroed out'. Mathematically, this idea can be represented by first partitioning the stiffness matrix in the following manner:

$$[\mathbf{K}] = \begin{bmatrix} \mathbf{K}^p_{aa} + \mathbf{K}^s_{aa} & \mathbf{K}^p_{ao} \\ \mathbf{K}^p_{oa} & \mathbf{K}^p_{oo} \end{bmatrix} \tag{25}$$

where the superscript p is used to indicate the plate and s refers to the springs. Let $\mathbf{x}$ be a displacement vector that imparts no internal strain energy. This now defines the a-set and the o-set. Further partitioning of Eq. (23) yields:

$$[\mathbf{K}] = \begin{bmatrix} \mathbf{K}^p_{aa} & \mathbf{K}^p_{ao} \\ \mathbf{K}^p_{oa} & \mathbf{K}^p_{oo} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_a \\ \mathbf{x}_o \end{Bmatrix} + \begin{bmatrix} \mathbf{K}^s_{aa} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_a \\ \mathbf{x}_o \end{Bmatrix} \tag{26}$$

Now, applying Eq. (24) to the general stiffness relation, $\mathbf{f} = \mathbf{kx}$, the following relation between the a-set and the o-set is derived much in the same manner

as Eq. (3):

$$\{x_o\} = -\left[K_{oo}^p\right]^{-1}\left[K_{oa}^p\right]\{x_a\} \tag{27}$$

By condensing the stiffness of the grounded springs out of the static transformation equation, the o-set exhibits the response of an unrestrained structure. The rigid body motion exhibited in Example (3) is a result of the stiffness matrix being partitioned in this manner.

A problem that affects the accuracy of both the static and IRS reduction methods is choosing the a-set system at points that are 'too close' together. In this situation, common sense must be depended upon because the concept of 'close' and 'far' depends on the size and geometry of the structure. Several simulations were conducted to study this and the best solution is simply to ensure the accelerometers are spread out to locations that best cover the entire structure. Having said this, it is important that the a-set include locations where the motion of the structure has the greatest amplitude. In the case of the plate model, the amplitude of the vibration is greatest at the center of the plate so the a-set should include time-history information from this area. A simulation was run where the a-set was comprised of edge nodes. The plate moves very little at its edges and subsequently the solution was inaccurate for both methods. Sound engineering judgment must be employed when choosing an a-set because it is critical to getting an accurate solution.

C. ROTATIONS AS PART OF THE A-SET

In general, using rotations, or a combination of rotations and translations as part of the a-set does not seem to alter the accuracy of the solution. The same rules apply for rotations as for the transverse displacements. A larger a-set still returns a more accurate solution and the location is still important. In this situation, however, being closer to the edge produced more accurate solutions. This makes sense because the edges are the locations of the largest rotations. Using rotations in the a-set is an advantage if applying the static reduction method because there is no longer a restriction on the use of nodes constrained through translational springs.

While using rotations is not a problem computationally, care must be taken to ensure that the structure is rotating along the axis corresponding to the chosen DOF. If not, including these DOF in the a-set serves no purpose as no strain energy is imparted into the system and the plate will again exhibit rigid body motion.

D. VISUALIZING A STRUCTURE

Animation in MATLAB is very computer memory intensive. A simple structure such as the plate can tax the memory of most computers and cause the program to crash. As such, a very large structure, or one with many nodes, would be almost impossible to visualize. To aid in overcoming this problem, the code should direct the computer to reduce the size of the graphics window for the animation. Although this does not allow for as nice a visualization, memory is saved for computation instead of being allocated

to the graphics window. By doing this, the visualization can be of longer duration thereby resulting in a better understanding of the dynamics of the structure.

A better tool to use in performing a dynamic visualization is I-DEAS. Unfortunately, certain versions will not allow the user to expand time histories. An outline can be found in the appendix that describes the steps needed to perform an animation in I-DEAS.

VI. CONCLUSIONS AND RECOMMENDATIONS

The main objective of this thesis was to investigate the process of visualizing transient structural response by expanding spatially incomplete time history data to full-order. The response at every node of a full-order plate model was solved which provided a base model for comparison. From this collection of nodal responses, a few of the time histories were extracted which simulated the experimental data taken from an actual structure. After choosing the a-set, the finite element system matrices were partitioned to calculate the appropriate transformation matrix. The expanded data was then compared to the full-order response in the effort of evaluating the accuracy of the process.

A. CONCLUSIONS

The simulations have shown the following:

1. To get an accurate solution, the minimum number of DOF's allowed to make up the a-set system on a flat plate is two. Provided the accelerometers are properly placed, an a-set comprised of four DOF will produce a very accurate response for both the static and the IRS methods.
2. The placement of the accelerometers is very important. If the static reduction method is chosen, the data must be taken at nodes certain to produce internal strain energy. Regardless of method, the data

must be extracted at locations 'far' enough from each other to get a sample of the entire structure, and the data collector should try to pick points on the structure where vibration is greatest.

3. The same restrictions that apply to using translations in the a-set can be applied to rotations.
4. The visualization process is very memory intensive therefore clever programming is required to maximize the memory available for graphics.

Visualizing transient response provides a unique insight on the motion of a vibrating structure. A much better understanding of the dynamics is gained when a designer can visualize the motion of a spatially complete body instead of the time histories of a few select nodes.

B. RECOMMENDATIONS

This study outlines a few of the strengths and weaknesses inherent to the Guyan and IRS reduction methods in the attempt to animate transient response. While many simulations were explored, the investigation was rather limited in scope. In addition to the Guyan and IRS methods, there are many reduction methods that could be explored. To validate the process, experimental data must be loaded into the program and studied. A similar study should be directed towards visualizing the response of three dimensional structures. In addition, it would be useful to investigate animating transient response in six degrees of freedom.

The accuracy of this process is heavily dependent on the location of the a-set. There are analytical techniques available that help find the 'best' a-set but ordinarily these are used when applying model reduction methods to find modeshapes. These techniques should to be investigated to prove if they can also be applied to this process.

The motivation for this thesis will be realized when the time history data of the DDG-51 class shock trial is loaded into I-DEAS, expanded to full-order, and then animated in a transient response visualization.

APPENDIX A. EXPANDING TIME HISTORIES IN I-DEAS

Step-by-step procedures for expanding transient time history data using I-DEAS software is provided. This option is not available using I-DEAS Master Series 3 if the software is set up to run on Hewlett-Packer computers.

A. ACCESS I-DEAS SIMULATION INTERFACE SOFTWARE

1. Master Modeler/Surfacing Task - load/build FE model
2. Meshing Task - mesh model and enter material properties
3. Boundary Condition Task -
    a. Choose normal mode dynamics
    b. Apply restraints
    c. Create boundary condition set/Choose normal mode dynamics - Guyan
    d. Choose analysis set (a-set)
4. Model Solution Task
    a. Create solution set
    b. Initiate solve
    c. Access I-DEAS model response software
5. Model Response Software
    a. Excitation definition
        - Create force/displacement excitation function in time domain

b. Response Evaluation

- Create response set

- initiate solve

c. Save transient response at a-set dof to a file called 'name.unv'

B. ACCESS I-DEAS TEST INTERFACE SOFTWARE

1. Correlation Task

a. import 'name.unv' file and change to an Attached Data File (ADF) called 'name.ash'

b. select mode shape to work with

- choose 'name.ash'

- select substructure component

2. Post-Processing Task

a. start animation

note: if using experimental data, once solve is initiated in the Simulation Interface Software, the Guyan transformation matrix is created. Proceed directly to Test Interface and load the .unv file of a-set time histories

APPENDIX B. MATLAB CODE

A. MAIN FE PROGRAM

```
%
%       Program will solve for the system mass and stiffness
%       matrices of a plate.  It uses four-node isoparametric
%       elements of the shear-deformable displacement
%       formulation.  The system mass and stiffness matrices
%       are saved to a file called sys_mat.mat
%
%                                                  Written by Scott Waltermire
%
%
%
%       Based on EX1071.m
%       Prof. Young Kwon
%
%
%
%--------------------------------------------------------------------------
%
%
% Variable descriptions
%  k = element matrix
%  kb = element matrix for bending stiffness
%  ks = element matrix for shear stiffness
%  f = element vector
%  kk = system matrix
%  ff = system vector
%  disp = system nodal displacement vector
%  gcoord = coordinate values of each node
%  nodes = nodal connectivity of each element
%  index = a vector containing system dofs associated with each element
%  pointb = matrix containing sampling points for bending term
%  weightb = matrix containing weighting coefficients for bending term
%  points = matrix containing sampling points for shear term
%  weights = matrix containing weighting coefficients for shear term
%  bcdof = a vector containing dofs associated with boundary conditions
%  bcval = a vector containing boundary condition values associated with
%          the dofs in 'bcdof'
%  kinmtpb = matrix for kinematic equation for bending
%  matmtpb = matrix for material property for bending
%  kinmtps = matrix for kinematic equation for shear
%  matmtps = matrix for material property for shear
%
%
%--------------------------------------------------------------------------
```

```
%-----------------------------------
%  input data for control parameters
%-----------------------------------
clear

hplate=100;                 % height of plate
lplate=100;                 % length of plate
hele=20;                    % element height
lele=20;                    % element length
a_ele=hele*lele;            % element area
nel=25;                     % number of elements
nnel=4;                     % number of nodes per element
ndof=3;                     % number of dofs per node
nnode=36;                   % total number of nodes in system
sdof=nnode*ndof;            % total system dofs
edof=nnel*ndof;             % degrees of freedom per element
emodule=30e6;               % elastic modulus
rho=0.284/386.4;            % density
poisson=0.3;                % Poisson's ratio
t=0.125;                    % plate thickness
nglxb=2; nglyb=2;           % 2x2 Gauss-Legendre quadrature for bending
nglb=nglxb*nglyb;           % number of sampling points per element for bending
nglxs=1; nglys=1;           % 1x1 Gauss-Legendre quadrature for shear
ngls=nglxs*nglys;           % number of sampling points per element for shear


%-------------------------------------------
%  input data for nodal coordinate values
%  gcoord(i,j) where i->node no. and j->x and y
%           size(gcoord) = num_nodes x 2
%
%       note:   if the elements used are not rectangular
%                       in shape, the values of gcoord and nodes
%                       must be manually entered by the user
%-------------------------------------------


count=0;
x = 0:lele:lplate;
y = 0:hele:hplate;
gnode = zeros(length(y),length(x));
gcoord=zeros(length(x)*length(y),2);


 for n=1:length(y)
   for j=1:length(x)
   count = count+1;
   gcoord(count,1:2) = gcoord(count,1:2) + [x(j) y(n)];
   gnode(n,j) = gnode(n,j)+count;
  end
 end
end
```

```
%
%       nodes is the nodal connectivity of the model
%      applied in the counter-clockwise direction
%

nodes=[];
count=0;
 for n=1:length(y)-1
  for j=1:length(x)-1
   count = count+1;
   node(count,1:4)=[gnode(n,j) gnode(n,j+1) gnode(n+1,j+1) gnode(n+1,j)];
   nodes = [nodes;node(count,1:4)];
  end
 end



%-------------------------------------------
% initialization of matrices and vectors
%-------------------------------------------

ff=zeros(sdof,1);              % system force vector
kk=zeros(sdof,sdof);           % system k matrix
mm = zeros(sdof,sdof);         % system m matrix
disp=zeros(sdof,1);            % system displacement vector
index=zeros(edof,1);           % index vector
kinmtpb=zeros(3,edof);         % kinematic matrix for bending
matmtpb=zeros(3,3);            % constitutive matrix for bending
kinmtps=zeros(2,edof);         % kinematic matrix for shear
matmtps=zeros(2,2);            % constitutive matrix for shear



%-------------------------------------------------------------
% computation of element matrices and vectors and their assembly
%-------------------------------------------------------------

%
% for bending stiffness
%

[pointb,weightb]=feglqd2(nglxb,nglyb);          % sampling points & weights
matmtpb=fematiso(1,emodule,poisson)*t^3/12;     % bending material property
```

```
%
%  for shear stiffness
%

[points,weights]=feglqd2(nglxs,nglys);        % sampling points & weights
shearm=0.5*emodule/(1.0+poisson);             % shear modulus
shcof=5/6;                                    % shear correction factor
matmtps=shearm*shcof*t*[1 0; 0 1];            % shear material property

for iel=1:nel                                 % loop for the total number of elements

for i=1:nnel
nd(i)=nodes(iel,i);                           % extract connected node for (iel)-th element
xcoord(i)=gcoord(nd(i),1);                    % extract x value of the node
ycoord(i)=gcoord(nd(i),2);                    % extract y value of the node
end

k=zeros(edof,edof);                           % initialize element stiffness matrix to zero
m=zeros(edof,edof);                           % initialize element mass matrix to zero
kb=zeros(edof,edof);                          % initialization of bending matrix to zero
ks=zeros(edof,edof);                          % initialization of shear matrix to zero


%-----------------------------------------------------
%  numerical integration for bending term
%-----------------------------------------------------

for intx=1:nglxb
x=pointb(intx,1);                             % sampling point in x-axis
wtx=weightb(intx,1);                          % weight in x-axis
for inty=1:nglyb
y=pointb(inty,2);                             % sampling point in y-axis
wty=weightb(inty,2) ;                         % weight in y-axis

[shape,dhdr,dhds]=feisoq4(x,y);               % compute shape functions and
                                              % derivatives at sampling point

jacob2=fejacob2(nnel,dhdr,dhds,xcoord,ycoord);  % compute Jacobian
detjacob=det(jacob2);                         % determinant of Jacobian
invjacob=inv(jacob2);                         % inverse of Jacobian matrix

[dhdx,dhdy]=federiv2(nnel,dhdr,dhds,invjacob);  % derivatives w.r.t.
                                  % physical coordinate

kinmtpb=fekinepb(nnel,dhdx,dhdy);             % bending kinematic matrix

%-----------------------------------------
%  compute bending and mass element matrix
%-----------------------------------------
```

```
kb=kb+kinmtpb'*matmtpb*kinmtpb*wtx*wty*detjacob;


%
%       Create a matrix of shape functions: Q
%       Ensure each row vector in the matrix is properly weighted: H(i)
%

Q = [shape(1) 0 0 shape(2) 0 0 shape(3) 0 0 shape(4) 0 0;...
        0 shape(1) 0 0 shape(2) 0 0 shape(3) 0 0 shape(4) 0;
        0 0 shape(1) 0 0 shape(2) 0 0 shape(3) 0 0 shape(4)];

H1 = shape(1)*Q;
H1 = [1/12*rho*t^3*H1(1,:);1/12*rho*t^3*H1(2,:);rho*t*H1(3,:)];
H2 = shape(2)*Q;
H2 = [1/12*rho*t^3*H2(1,:);1/12*rho*t^3*H2(2,:);rho*t*H2(3,:)];
H3 = shape(3)*Q;
H3 = [1/12*rho*t^3*H3(1,:);1/12*rho*t^3*H3(2,:);rho*t*H3(3,:)];
H4 = shape(4)*Q;
H4 = [1/12*rho*t^3*H4(1,:);1/12*rho*t^3*H4(2,:);rho*t*H4(3,:)];

%       Create the elemental mass matrix in terms of shape functions

H = [H1;H2;H3;H4];

%
%       Solve for the elemental mass matrix
%

m = m+H*wtx*wty*detjacob;

end
end             % end of numerical integration loop for bending term
                                        % and mass term


%-------------------------------------------------------
%  numerical integration for shear term
%-------------------------------------------------------

for intx=1:nglxs
x=points(intx,1);                       % sampling point in x-axis
wtx=weights(intx,1);                    % weight in x-axis
for inty=1:nglys
y=points(inty,2);                       % sampling point in y-axis
wty=weights(inty,2) ;                   % weight in y-axis

[shape,dhdr,dhds]=feisoq4(x,y);         % compute shape functions and
                                        % derivatives at sampling point

jacob2=fejacob2(nnel,dhdr,dhds,xcoord,ycoord);  % compute Jacobian
```

```
detjacob=det(jacob2);                          % determinant of Jacobian
invjacob=inv(jacob2);                          % inverse of Jacobian matrix

[dhdx,dhdy]=federiv2(nnel,dhdr,dhds,invjacob);  % derivatives w.r.t.
                                  % physical coordinate

kinmtps=fekineps(nnel,dhdx,dhdy,shape);         % shear kinematic matrix


%---------------------------------------
%  compute shear element matrix
%---------------------------------------

ks=ks+kinmtps'*matmtps*kinmtps*wtx*wty*detjacob;

end
end              % end of numerical integration loop for shear term

%-------------------------------
%  compute element matrix
%-------------------------------

k=kb+ks;


index=feeldof(nd,nnel,ndof);        % extract system dofs associated with element

kk=feasmbl1(kk,k,index);            % assemble element matrices into system
mm=feasmbl1(mm,m,index);            % matrices

end


% Apply Springs in z-direction

kk(3,3) = kk(3,3) + 100;
kk(18,18) = kk(18,18) + 100;
kk(93,93) = kk(93,93) + 100;
kk(108,108) = kk(108,108) + 100;

save sys_mat mm kk
```

B. FUNCTIONS CALLED BY MAIN FE PROGRAM

```
function [point2,weight2]=feglqd2(nglx,ngly)

%-------------------------------------------------------------------
% Purpose:
%    determine the integration points and weighting coefficients
%    of Gauss-Legendre quadrature for two-dimensional integration
%
% Synopsis:
%    [point2,weight2]=feglqd2(nglx,ngly)
%
% Variable Description:
%    nglx - number of integration points in the x-axis
%    ngly - number of integration points in the y-axis
%    point2 - vector containing integration points
%    weight2 - vector containing weighting coefficients
%-------------------------------------------------------------------

% determine the largest one between nglx and ngly

  if nglx > ngly
    ngl=nglx;
  else
    ngl=ngly;
  end

% initialization

  point2=zeros(ngl,2);
  weight2=zeros(ngl,2);

% find corresponding integration points and weights

 [pointx,weightx]=feglqd1(nglx);      % quadrature rule for x-axis
 [pointy,weighty]=feglqd1(ngly);      % quadrature rule for y-axis

% quadrature for two-dimension

 for intx=1:nglx                      % quadrature in x-axis
   point2(intx,1)=pointx(intx);
   weight2(intx,1)=weightx(intx);
 end

 for inty=1:ngly                      % quadrature in y-axis
   point2(inty,2)=pointy(inty);
   weight2(inty,2)=weighty(inty);
 end
```

```
function [matmtrx]=fematiso(iopt,elastic,poisson)

%-----------------------------------------------------------------------
% Purpose:
%    determine the constitutive equation for isotropic material
%
% Synopsis:
%    [matmtrx]=fematiso(iopt,elastic,poisson)
%
% Variable Description:
%    elastic - elastic modulus
%    poisson - Poisson's ratio
%    iopt=1 - plane stress analysis
%    iopt=2 - plane strain analysis
%    iopt=3 - axisymmetric analysis
%    iopt=4 - three dimensional analysis
%-----------------------------------------------------------------------

if iopt==1       % plane stress
  matmtrx= elastic/(1-poisson*poisson)* ...
  [1  poisson 0; ...
  poisson  1  0; ...
  0  0  (1-poisson)/2];

elseif iopt==2    % plane strain
  matmtrx= elastic/((1+poisson)*(1-2*poisson))* ...
  [(1-poisson)  poisson 0;
  poisson  (1-poisson)  0;
  0  0  (1-2*poisson)/2];

elseif iopt==3    % axisymmetry
  matmtrx= elastic/((1+poisson)*(1-2*poisson))* ...
  [(1-poisson)  poisson  poisson  0;
  poisson  (1-poisson)  poisson  0;
  poisson  poisson  (1-poisson)  0;
  0   0   0   (1-2*poisson)/2];

else    % three-dimension
  matmtrx= elastic/((1+poisson)*(1-2*poisson))* ...
  [(1-poisson)  poisson  poisson  0   0   0;
  poisson  (1-poisson)  poisson  0   0   0;
  poisson  poisson  (1-poisson)  0   0   0;
  0   0   0   (1-2*poisson)/2  0   0;
  0   0   0   0   (1-2*poisson)/2  0;
  0   0   0   0   0   (1-2*poisson)/2];

end
```

```
function [shapeq4,dhdrq4,dhdsq4]=feisoq4(rvalue,svalue)

%-------------------------------------------------------------------
% Purpose:
%    compute isoparametric four-node quadilateral shape functions
%    and their derivatves at the selected (integration) point
%    in terms of the natural coordinate
%
% Synopsis:
%    [shapeq4,dhdrq4,dhdsq4]=feisoq4(rvalue,svalue)
%
% Variable Description:
%    shapeq4 - shape functions for four-node element
%    dhdrq4 - derivatives of the shape functions w.r.t. r
%    dhdsq4 - derivatives of the shape functions w.r.t. s
%    rvalue - r coordinate value of the selected point
%    svalue - s coordinate value of the selected point
%
% Notes:
%    1st node at (-1,-1), 2nd node at (1,-1)
%    3rd node at (1,1), 4th node at (-1,1)
%-------------------------------------------------------------------

% shape functions

shapeq4(1)=0.25*(1-rvalue)*(1-svalue);
shapeq4(2)=0.25*(1+rvalue)*(1-svalue);
shapeq4(3)=0.25*(1+rvalue)*(1+svalue);
shapeq4(4)=0.25*(1-rvalue)*(1+svalue);

% derivatives

dhdrq4(1)=-0.25*(1-svalue);
dhdrq4(2)=0.25*(1-svalue);
dhdrq4(3)=0.25*(1+svalue);
dhdrq4(4)=-0.25*(1+svalue);

dhdsq4(1)=-0.25*(1-rvalue);
dhdsq4(2)=-0.25*(1+rvalue);
dhdsq4(3)=0.25*(1+rvalue);
dhdsq4(4)=0.25*(1-rvalue);
```

```
function [jacob2]=fejacob2(nnel,dhdr,dhds,xcoord,ycoord)

%-------------------------------------------------------------------
% Purpose:
%    determine the Jacobian for two-dimensional mapping
%
% Synopsis:
%    [jacob2]=fejacob2(nnel,dhdr,dhds,xcoord,ycoord)
%
% Variable Description:
%    jacob2 - Jacobian for one-dimension
%    nnel - number of nodes per element
%    dhdr - derivative of shape functions w.r.t. natural coordinate r
%    dhds - derivative of shape functions w.r.t. natural coordinate s
%    xcoord - x axis coordinate values of nodes
%    ycoord - y axis coordinate values of nodes
%-------------------------------------------------------------------

jacob2=zeros(2,2);

for i=1:nnel
jacob2(1,1)=jacob2(1,1)+dhdr(i)*xcoord(i);
jacob2(1,2)=jacob2(1,2)+dhdr(i)*ycoord(i);
jacob2(2,1)=jacob2(2,1)+dhds(i)*xcoord(i);
jacob2(2,2)=jacob2(2,2)+dhds(i)*ycoord(i);
end


function [dhdx,dhdy]=federiv2(nnel,dhdr,dhds,invjacob)

%-------------------------------------------------------------------
% Purpose:
%    determine derivatives of 2-D isoparametric shape functions with
%    respect to physical coordinate system
%
% Synopsis:
%    [dhdx,dhdy]=federiv2(nnel,dhdr,dhds,invjacob)
%
% Variable Description:
%    dhdx - derivative of shape function w.r.t. physical coordinate x
%    dhdy - derivative of shape function w.r.t. physical coordinate y
%    nnel - number of nodes per element
%    dhdr - derivative of shape functions w.r.t. natural coordinate r
%    dhds - derivative of shape functions w.r.t. natural coordinate s
%    invjacob - inverse of 2-D Jacobian matrix
%-------------------------------------------------------------------



for i=1:nnel
dhdx(i)=invjacob(1,1)*dhdr(i)+invjacob(1,2)*dhds(i);
dhdy(i)=invjacob(2,1)*dhdr(i)+invjacob(2,2)*dhds(i);
end
```

```
function [kinmtpb]=fekinepb(nnel,dhdx,dhdy)

%-------------------------------------------------------------------
% Purpose:
%    determine the kinematic matrix expression relating bending curvatures
%    to rotations and displacements for shear deformable plate bending
%
% Synopsis:
%    [kinmtpb]=fekinepb(nnel,dhdx,dhdy)
%
% Variable Description:
%    nnel - number of nodes per element
%    dhdx - derivatives of shape functions with respect to x
%    dhdy - derivatives of shape functions with respect to y
%-------------------------------------------------------------------

for i=1:nnel
i1=(i-1)*3+1;
i2=i1+1;
i3=i2+1;
kinmtpb(1,i1)=dhdx(i);
kinmtpb(2,i2)=dhdy(i);
kinmtpb(3,i1)=dhdy(i);
kinmtpb(3,i2)=dhdx(i);
kinmtpb(3,i3)=0;
end


function [kk]=feasmbl1(kk,k,index)
%-----------------------------------------------------
% Purpose:
%    Assembly of element matrices into the system matrix
%
% Synopsis:
%    [kk]=feasmbl1(kk,k,index)
%
% Variable Description:
%    kk - system matrix
%    k  - element matri
%    index - d.o.f. vector associated with an element
%-----------------------------------------------------


edof = length(index);
for i=1:edof
  ii=index(i);
    for j=1:edof
      jj=index(j);
        kk(ii,jj)=kk(ii,jj)+k(i,j);
    end
end
```

```
function [kinmtps]=fekineps(nnel,dhdx,dhdy,shape)

%-------------------------------------------------------------------
% Purpose:
%    determine the kinematic matrix expression relating shear strains
%    to rotations and displacements for shear deformable plate bending
%
% Synopsis:
%    [kinmtps]=fekineps(nnel,dhdx,dhdy,shape)
%
% Variable Description:
%    nnel - number of nodes per element
%    dhdx - derivatives of shape functions with respect to x
%    dhdy - derivatives of shape functions with respect to y
%    shape - shape function
%-------------------------------------------------------------------

for i=1:nnel
i1=(i-1)*3+1;
i2=i1+1;
i3=i2+1;
kinmtps(1,i1)=-shape(i);
kinmtps(1,i3)=dhdx(i);
kinmtps(2,i2)=-shape(i);
kinmtps(2,i3)=dhdy(i);
end

function [index]=feeldof(nd,nnel,ndof)
%-----------------------------------------------------------
% Purpose:
%    Compute system dofs associated with each element
%
% Synopsis:
%    [index]=feeldof(nd,nnel,ndof)
%
% Variable Description:
%    index - system dof vector associated with element "iel"
%    nd - nodal vector whose system dofs are to be determined
%    nnel - number of nodes per element
%    ndof - number of dofs per node
%-----------------------------------------------------------

edof = nnel*ndof;
  k=0;
  for i=1:nnel
    start = (nd(i)-1)*ndof;
      for j=1:ndof
        k=k+1;
        index(k)=start+j;
      end
  end
```

C. MAIN POST-PROCESSING PROGRAM

```
clear

%-------------------------------------------------------------
%
%       This program will load the system mass and stiffness matrices
%       from file sys_mat.mat.  It solves the eigenvalue problem
%       to get the natural frequencies and the modeshapes.
%       Convolution is used to solve for the modal solution q(t).
%       It solves for x(t) at all nodes then calls set.mat to
%       pick out the response at the analysis set.  From this it
%       back-expands the aset into the full solution using Guyan and
%       IRS transformation matrices.  It can plot the modeshapes,
%       time-histories of each solution and the error between
%       the actual response and the Guyan/IRS models.
%
%                                                   Written by Scott Waltermire

% Get the global mass and stiffness matrices

load sys_mat;
sdof=108;

% Create Blast Forcing Function

plotit = 1;                                 % plotit=1 plots forcing function
Fo = 2000;                                  % amplitude of forcing function
time = 0:.01:3;                             % time step
[f_of_t] = fBlastForcing(Fo,time,plotit);   % function to solve for f(t)
F = zeros(sdof,length(time));               % initialize system force vector
F(63,:) = f_of_t;                           % place f(t) at the proper node


%  Solve and Sort Eigenvalues/Modeshapes

[phi,lam]=eig(mm\kk);
mtilda=phi'*mm*phi;                         % mass normalize

% loop to mass normalize the modeshapes

 for i=1:length(mm)
  phi(:,i)=phi(:,i)*1/(sqrt(mtilda(i,i)));
 end

% loop to pick out the natural frequencies

 for j=1:length(mm)
  ev(j)=lam(j,j);
 end

[lam,p]=sort(ev);                           % sort the natural frequencies
```

```
phistar=phi;

% loop to sort the eigenvectors in the same manner
% as the natural frequencies

 for k=1;length(mm)
  phi(:,k)=phistar(:,p(k));
 end

wn=real(sqrt(lam));
omega=real((sqrt(lam)/(2*pi)))';                    % convert wn to hertz
phi = real(phi);

%  create modal_F and use proportional damping

modal_F = phi'*F;
zeta = .02;

% Convolution to solve for q(t) using Trapezoidal rule

dt = .001;
for i = 1:108
 wd(i) = wn(i)*sqrt(1-zeta^2);
 h = 1/wd(i)*exp(-zeta*wn(i)*time).*sin(wd(i)*time);
 F = modal_F(i,:);
 [convXY] = fconvTrap(F,h,dt);
 q(i,:) = convXY;
end

% Plot Modeshapes

num_mode_shape=10;
 for i=1:num_mode_shape
   phi_mode=real(phi(:,i));
   phi1=(phi_mode(3:3:18))';
   phi2=(phi_mode(21:3:36))';
   phi3=(phi_mode(39:3:54))';
   phi4=(phi_mode(57:3:72))';
   phi5=(phi_mode(75:3:90))';
   phi6=(phi_mode(93:3:108))';
   phi_mesh=[phi1;phi2;phi3;phi4;phi5;phi6];
   surfc(phi_mesh)
   shading interp
   grid
   pause
 end
```

```
% Convert back to physical Coordinates

resp = phi*q;

% Solve for T_static and T_irs

load set;                                          % aset/oset file

%       functions to solve for Guyan and IRS
%       transformation matrices.

[kstat,mstat,T_static] = fstatic_tam(kk,mm,oset,aset);
[kirs,mirs,T_irs] = firs_tam(kk,mm,oset,aset);


%-------------------------------------------
% Load Guyan Transformation matrix
% and determine xa and xo
%-------------------------------------------

 for i = 1:length(aset)
   xa_g(i,:) = resp(aset(i),:);
 end

% Back expand the guyan aset time histories
% into the full response

x_g = T_static*xa_g;
xo_g = x_g(1:sdof-length(aset),:);
xa_g = x_g(sdof-length(aset)+1:sdof,:);

% Now re-sort xa_g and xo_g

 for i=1:length(aset)
   xo1_g = xo_g(1:aset(i)-1,:);
   xo2_g = xo_g(aset(i):length(xo_g(:,1)),:);
   xo1_g = [xo1_g;xa_g(i,:)];
   xo_g = [xo1_g;xo2_g];
 end
x_g = xo_g;


%-----------------------------------------
% Load IRS Transformation matrix
% and determine xa and xo
%-----------------------------------------


 for i = 1:length(aset)
   xa_irs(i,:) = resp(aset(i),:);
 end
```

```
% Back expand the irs aset time histories
% into the full response

x_irs = T_irs*xa_irs;
xo_irs = x_irs(1:sdof-length(aset),:);
xa_irs = x_irs(sdof-length(aset)+1:sdof,:);

% Now re-sort xa_irs and xo_irs

 for i=1:length(aset)
   xo1_irs = xo_irs(1:aset(i)-1,:);
   xo2_irs = xo_irs(aset(i):length(xo_irs(:,1)),:);
   xo1_irs = [xo1_irs;xa_irs(i,:)];
   xo_irs = [xo1_irs;xo2_irs];
 end
x_irs = xo_irs;


% Ignore all rotations and plot response
% in the z-direction

resp = resp(3:3:sdof,:);
resp_g = x_g(3:3:sdof,:);
resp_irs = x_irs(3:3:sdof,:);

for i=1:36

 figure(1)
   plot(time,resp(i,:),'r-.',time,resp_g(i,:),'y');
  xlabel('Time (sec)');
  ylabel('Displacement (in)');
  title('Time History');

 figure(2)
   plot(time,resp(i,:),'r-.',time,resp_irs(i,:),'y');
  xlabel('Time (sec)');
  ylabel('Displacement (in)');
  title('Time History');

 pause
end

close(1)
close(2)
```

```
%-----------------------------------------------
%       Find the error between the true response and
%       the Guyan/IRS back-expanded response
%-----------------------------------------------

for i = 1:length(time)
 xg_err(:,i) = abs(resp_g(:,i) - resp(:,i));
 xirs_err(:,i) = abs(resp_irs(:,i) - resp(:,i));
end
for i = 1:36
 xg_error(i) = max(xg_err(i,:));
 xirs_error(i) = max(xirs_err(i,:));
end

xg_error1 = xg_error(1:6);
xg_error2 = xg_error(7:12);
xg_error3 = xg_error(13:18);
xg_error4 = xg_error(19:24);
xg_error5 = xg_error(25:30);
xg_error6 = xg_error(31:36);
xg_error = [xg_error1;xg_error2;xg_error3;xg_error4;xg_error5;xg_error6];
figure(1)
surf(xg_error);
shading interp
grid

xirs_error1 = xirs_error(1:6);
xirs_error2 = xirs_error(7:12);
xirs_error3 = xirs_error(13:18);
xirs_error4 = xirs_error(19:24);
xirs_error5 = xirs_error(25:30);
xirs_error6 = xirs_error(31:36);
xirs_error = [xirs_error1;xirs_error2;xirs_error3;...
                        xirs_error4;xirs_error5;xirs_error6];
figure(2)
surf(xirs_error);
shading interp
grid



save modal_info resp resp_g resp_irs time
```

D. FUNCTIONS CALLED BY MAIN PROGRAM

```
function [f_of_t] = fBlastForcing(Fo,time,plotit);
%
%   This function returns a forcing function which is
%   a "blast" function.
%
%       F(t) = Fo * ( exp(-at) - exp(-bt) )
%
%   where a and b are constants which shape the blast,
%   and Fo is the amplitude of the blast.
%
%   The variable "plotit" is a switch which if set = 1 will
%   cause the f(t) to be plotted, if set to anything else
%   will not plot.
%                                          written by J.H. Gordis
% ____________________________________________________________
% Choices: sine   blst    step

  type = 'blst';
% ~~~~~~~~~~~~~~

if type == 'blst';

     disp(' Blast forcing used...')
    a = 100.0;
    b = 300.0;
    f_of_t = Fo * ( exp(-a*time) - exp(-b*time) );

elseif type == 'step';

%     disp(' Step forcing used...')
    f_of_t = Fo * ones(size(time));

elseif type == 'sine';

%     disp(' Sine forcing used...')
    W = 5; % Hertz
    f_of_t = Fo * sin(2*pi*W*time);

end;

if plotit == 1;
  figure(9)
  plot(time,f_of_t);grid
  pause
  clf
end

% End function.
```

```
function [convXY] = fConvTrap(x,y,dt);
%
% Usage:   [convXY] = fConvTrap(x,y,dt);
%
% This function performs the convolution integral calculation
%
%                tau=t
%    convXY(t) =   S    x(t-tau) y(tau) d tau
%                tau=0
%
% using the trapezoidal rule.
% The function is passed the vectors x(t) and y(t)
% and the sample spacing (time step) dt.
%
%   x     = vector of length n
%   y     = vector of length n
%   dt    = sample spacing (i.e. delta_t).
%                                                    written by J.H. Gordis
% ____________________________________________________________

if size(x) == size(y);        % Vectors the same size. Perform convolution.
                    % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   convXY = zeros(size(x),1);

   for icnt_step = 2 : length(x);
     [wts] = fTrapzWts(icnt_step);
           if size(x,1) ~= size(wts,1);
              wts = wts';
           end
      convXY(icnt_step) = dt * sum(wts .* fliplr(x(1:icnt_step)) .* y(1:icnt_step) );
   end;

elseif size(x) ~= size(y) & length(x) == length(y);   % Vectors are transposed. Don't
perform convolution.
                                        %
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

         disp('  Error in fConvTrap. Transpose one vector and try again.')
                           disp(sprintf('          Size(x) %3i',size(x)))
         disp(sprintf('          Size(y) %3i',size(y)))

elseif size(x) ~= size(y) & length(x) ~= length(y);   % Vectors different sizes. Don't
perform convolution.
                                        %
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   disp(' Error in fConvTrap. Vector not the same size.')
                           disp(sprintf('          Size(x) %3i',size(x)))
         disp(sprintf('          Size(y) %3i',size(y)))

end;
```

```
%
function [kstat,mstat,T_static]=fstatic_tam(k,m,oset,aset)
%
% this function returns the statically reduced stiffness
% and mass matrices, given the unreduced couterparts.
% Care must be taken that the aset and oset vectors correspond
% with the existing arrangement of k and m.
% k and m are UNPARTITIONED matrices.
%
%                                                written by J.H. Gordis
aset_size=length(aset);
%
kaa=k(aset,aset);
kao=k(aset,oset);
koo=k(oset,oset);
koa=kao';
clear k;
k=[koo,koa;kao,kaa];
%
maa=m(aset,aset);
mao=m(aset,oset);
moo=m(oset,oset);
moa=mao';
clear m;
m=[moo,moa;mao,maa];
%
t_static=-koo\koa;
T_static = [t_static;eye(aset_size)];
%
kstat=T_static'*k*T_static;
mstat=T_static'*m*T_static;
%
% end function fstatic_tam
```

```
%
function [kirs,mirs,T_irs]=firs_tam(k,m,oset,aset)
%
% this function returns the IRS reduced stiffness
% and mass matrices, given the unreduced couterparts.
% Care must be taken that the aset and oset vectors correspond
% with the existing arrangement of k and m.
% k and m are UNPARTITIONED matrices.
%                                                    written by J.H. Gordis
aset_size=length(aset);
%
kaa=k(aset,aset);
kao=k(aset,oset);
koo=k(oset,oset);
koa=kao';
clear k;
k=[koo,koa;kao,kaa];
%
maa=m(aset,aset);
mao=m(aset,oset);
moo=m(oset,oset);
moa=mao';
clear m;
m=[moo,moa;mao,maa];
%
t_static=-koo\koa;
T_static = [t_static;eye(aset_size)];
%
kstat=T_static'*k*T_static;
mstat=T_static'*m*T_static;
%
tirs=t_static+inv(koo)*(moa+moo*t_static)*inv(mstat)*kstat;
T_irs=[tirs;eye(aset_size)];
%
kirs=T_irs'*k*T_irs;
mirs=T_irs'*m*T_irs;
%
% end function firs_tam
```

```
%-----------------------------------------------------------------------
%
%       File holding the dof for the analysis set (aset)
%       and the omitted set (oset)
%
%       Saved to a file called set.mat
%
%-----------------------------------------------------------------------

aset = [3 18 93 108];
oset = [1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 ...
        19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 ...
        40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ...
        62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 ...
        83 84 85 86 87 88 89 90 91 92 94 95 96 97 98 99 100 101 102 ...
        103 104 105 106 107];

save set aset oset
```

```
%       Program will animate the full-order response
%       by loading a .mat file called modal-info

clear;
load modal_info;
clear resp_g resp_irs

% make movie

movie_fig = figure('position',[100 200 300 200]);
M = moviein(200);
[x,y] = meshgrid([-5:2:5]);
 for i = 1:200
  resp_t = resp(:,i);
  resp1 = (resp_t(1:6))';
  resp2 = (resp_t(7:12))';
  resp3 = (resp_t(13:18))';
  resp4 = (resp_t(19:24))';
  resp5 = (resp_t(25:30))';
  resp6 = (resp_t(31:36))';
  z = [resp1;resp2;resp3;resp4;resp5;resp6];
surf(x,y,z);
shading interp
grid;
axis([-5 5 -5 5 -.4 .6]);
view([45 45 10])
zlabel('disp (in.)');
M(:,i) = getframe;
end
pause
movie(M,0);
```

```
%       Program will animate the plate response found
%       by the static reduction method. It loads the .mat
%       file called modal_info

clear;
load modal_info;

clear resp resp_irs
resp = resp_g;

% make movie of guyan response

movie_fig = figure('position',[100 200 300 200]);
M = moviein(200);
[x,y] = meshgrid([-5:2:5]);

count = 1;
 for i = 1:200
   resp_t = resp(:,i);
   resp1 = (resp_t(1:6))';
   resp2 = (resp_t(7:12))';
   resp3 = (resp_t(13:18))';
   resp4 = (resp_t(19:24))';
   resp5 = (resp_t(25:30))';
   resp6 = (resp_t(31:36))';
   z = [resp1;resp2;resp3;resp4;resp5;resp6];
surf(x,y,z);
shading interp
grid;
axis([-5 5 -5 5 -.4 .6]);
view([45 45 10])
zlabel('disp (in.)');
title('Animation by Guyan Back Expansion');
M(:,count) = getframe;
count = count+1;
end

pause

movie(M,0);
```

```
%
%       Program to animate the plate response found
%       by the IRS reduction method. Loads the .mat file
%       modal_info
%

clear;
load modal_info;

clear resp resp_g
resp = resp_irs;

% make movie of guyan response

movie_fig = figure('position',[100 200 300 200]);
M = moviein(200);
[x,y] = meshgrid([-5:2:5]);

count = 1;
  for i = 1:200
    resp_t = resp(:,i);
    resp1 = (resp_t(1:6))';
    resp2 = (resp_t(7:12))';
    resp3 = (resp_t(13:18))';
    resp4 = (resp_t(19:24))';
    resp5 = (resp_t(25:30))';
    resp6 = (resp_t(31:36))';
    z = [resp1;resp2;resp3;resp4;resp5;resp6];
surf(x,y,z);
shading interp
grid;
axis([-5 5 -5 5 -.4 .6]);
view([45 45 10])
zlabel('disp (in.)');
title('Animation by IRS Back Expansion');
M(:,count) = getframe;
count = count+1;
end

pause

movie(M,0);
```

LIST OF REFERENCES

1. Guyan, R.J. "Reduction of Mass and Stiffness Matrices," *AIAA Journal,* v.3 February, 1965 pp. 380.
2. O'Callahan, J. "A Procedure for an Improved Reduced System (IRS) Model". *Proceedings of the 7th International Modal Analysis Conference,* Las Vegas, NV, 1989, pp. 17-21.
3. Gordis, J.H., "An Analysis of the Improved Reduced System (IRS) Model Reduction Procedure". *Procededings of the 9th International Modal Analysis Conference,* 1994, pp. 271-273.
4. Kwon, Y.W., Bang, H., *The Finite Element Method Using MATLAB,* pp. 160-161, 366-369, CRC Press Inc., 1997.
5. Craig, R.R., *Structural Dynamics an Introduction to Computer Methods,* pp. 123, 207, 409, John Wiley & Sons, 1981.
6. I-DEAS Master Series 2.1 Manuals.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center 8725 John J. Kingman Road., Ste 0944 Ft. Belvoir, Va 22060-6218	2
2. Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, California 93943-5101	2
3. Professor J.H. Gordis, Code ME/Go Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1
4. Department Chairman, Code ME Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1
5. Naval Engineering Curricular Office (Code 34) Naval Postgraduate School Monterey, California 93943	1
6. LT Scott W. Waltermire Rt 2 Box 157 Perry, Oklahoma 73077	1